MOTOROLA SOLUTIONS

EWB 100 Usage and Deployment Guide

Rev 1.7

June 4, 2012

Revision History

Revision	Description	Editor
1.7	Release for Publication	Linda Billhymer

1 Introduction

1.1 *The EWB100*

The EWB100 is a compact Voice over wireless LAN (VoWLAN) communications device designed to operate throughout the wireless local area network of an enterprise environment. The EWB 100 provides Push-to-Talk style communications, similar to any normal Push-to-Talk radio. EWB100 badges enable instant employee accessibility through a simultaneous call to a group of workers and a person-to-person reply call from a group member. They can communicate with other Motorola mobile computers and enterprise devices communicating over Wi-Fi.

The EWB 100 can be clipped onto clothing, worn on a lanyard, or slipped inside a pocket and can be deployed either as a personal or shared device. In environments where workers change shifts daily, an employee can select any available badge.

1.2 *This Document*

This document is intended for those who select, configure, deploy, and support the EWB100 device. It is not directed at those who are the end users of it. The *EWB100 Quick Start Guide* is intended for such users.

This document describes the capabilities of the EWB100 and how the device can be configured to take advantage of those capabilities. While the EWB100 is typically used for its “Walkie talkie” like functions it is capable of much more and this document describes those capabilities.

This document also describes how the device can be deployed and updated over time. It does not describe the “Deployment Application” but instead describes how many of the same functions can be achieved with existing network software packages.

Also included in this document is a description of the mechanisms that are available to diagnose and fix problems that may arise during deployment or in an installation. The EWB100 has a number of powerful mechanisms that will allow on-site or remote maintenance.

Finally this document will describe enough of the internal architecture so as to make the process of configuring, optimizing, and supporting the device an easier task.

2 EWB100 User Overview

2.1 *The Physical Device*

The image below shows the EWB100 with the important user elements identified

2.2 *The User Interface*

The user interface of the EWB100 is very simple, consisting of 5 buttons,a tricolor LED, a microphone, and a speaker. There are also optional headsetsthat can substitute for the microphone and/or speaker. There is not a screen of any type. This means that most of the user interactions take the form of audio (mostly output) and button presses, perhaps supplemented by LED blink patterns.

Furthermore two of the buttons (VOLUP and VOLDN) are usually reserved for volume control functions (although they can be used for certain other tasks). That leaves 3 application oriented buttons: PTT, FAPP, and SAPP. The PTT button is used almost exclusively as a “”push to talk” trigger;there are really only two “application oriented” buttons.

The EWB gets around this limitation by two mechanisms: the first is to define certain button actions beyond just press and release. The first is a “single click”, which is a quick press and release. The second is a “double click” which is two quick press and release sequences. The third is a sustained press and hold. The EWB 100 treats each of these as a distinct event from a UI perspective.

The second mechanism is by multiple button sequences. At this time only two button sequences are used but potentially there could be three and four button sequences but they would be very awkward to use. There are currently 3 two button sequences defined:

- PTT + FAPP => Power off
- PTT + SAPP => Status/configuration
- VOLDN + SAPP (while in the cradle) => set to factory defaults

The LEDs are used mostly to indicate general device state. Blinking Green means everything is working properly. Blinking Red means something is wrong. Amber is not used at the moment.

Most of the use oriented “feedback” to the user is via the speaker. The EWB 100 uses a combination of tones and voice responses to indicate state and events to the user. All of these tones and voice responses can be replaced with silence or different audio sequences, including different languages.

2.3 *Voice Modes*

There are two primary voice modes. One is a walkie talkie (one to many) type. One user presses the PTT button and talks and lots of people hear. Just like more traditional walkie talkies, there are multiple channels available and users can pick which ones they want to use by switching between them.

The second mode is for private point to point calls. This is called *Private Reply* and keeps the same “push to talk” modality of the walkie talkie mode except that it is a private session between two users. Such a session is setup after the conclusion of a walkie talkie session via a simple button press sequence.

There are other voice modes beyond these two that will be described later in this document.

3 Internal Elements of the EWB100

This section introduces some of the more important internal aspects of the EWB100 as well as some features beyond just the voice modes mentioned in the previous section.

3.1 *Two software subsystems*

There are two software systems that run on the EWB100 device. The first is the **"**mini-kernel**"** or "boot loader" which runs right after power on or reset. This system does not have a lot of functionality but exists mostly to handle the reloading of the main system when needed. This software has a limited set of commands available but for the most part cannot examine or alter the system configuration (although it can erase it entirely!). The minikernel's commands are documented in a separate section of this document

The second software system is the actual runtime code that has lots of functionality and an extensive set of commands. Most of the commands in this document apply only to the runtime software.

3.2 *Configuration Data Base*

There are many configuration parameters for the EWB100 device. Almost all aspects of the behavior of the device may be altered by configuration parameters. All configuration parameters have a default value that is part of the runtime image and cannot be changed by the user except for some limited values via the "User Defined Defaults" mechanism.

3.3 *802.11 Radio*

The EWB100 contains an 802.11 radio that operates in the 2.4GHz band. It supports both 802.11b and 802.11g. The radio has been highly optimized to reduce power consumption both when operating and when in standby mode.

3.4 *IP Stack*

The EWB100 contains an IPv4 protocol stack. It supports DHCP for obtaining IP addresses. It also includes a simple Telnet server that can be used for diagnostic and maintenance purposes.

3.5 *Command Line Interpreter*

There is a command line interpreter in the EWB100 that provides a means to configure, update, monitor, and diagnose it. It provides complete access to the configuration data base and so is used whenever the device needs to be configured.

There are two versions of the CLI, one for the miniKernel and one for the runtime code.

3.6 *Profiles*

A **profile** is a set of configuration variables that control the user interface and overall functionality of a EWB100 device. Different profiles are simply different sets of the same configuration variables. Profiles define what functions are associated with particular buttons, the alerts and notifications associated with UI events, walkie talkie channels, etc. Virtually all user visible configuration parameters are part of a profile. There are current 8 profiles per EWB100 and they are accessed by selecting from profile numbers 0 to 7. Users select the desired profile from the user interface.

3.7 *Airbeam and MSP*

Airbeam may be used to update the runtime code and configuration elements of the EWB100. All updates are performed over the 802.11 network using the FTP network protocol. Updates are described in files called “packages” and the EWB looks for new packages on a regular basisThe EWB 100 also supports MSP for runtime code updates and configuration.

3.8 *Audio Clips and Default Dictionary*

The EWB does not contain a text to speech engine but rather has a dictionary of audio files called ”clips”. The device combines these clips into a meaningful verbal phrase as needed. Some clips are tones while others are words. There are about 500 audio clips stored in the “default dictionary” of the device by default. New clips can be added to the device to enhance or replace the contents of the default dictionary. These clips may be in other languages. The Default Dictionary is contained in Appendix E.

3.9 *Web Server*

The EWB100 also contains a simple web server. External applications can use this web server to command the device to perform simple tasks such as to play one or more audio clips, flash the LEDs, and reset the device. The CLI is also available using the Web Server**.**

4 The Command Line Interpreter

4.1 *Basic Concepts*

The CLI on the EWB100 is used for a wide variety of functions including configuration, updates, diagnostics, and maintenance. Its primary use, though, is for configuring the device. The interface is very simple; it features one line commands consisting of a key word and zero or more parameters. Most commands take effect immediately.

The CLI is always available on the device and can be reached from a variety of sources:

- USB
- Telnet
- HTTP
- Command Files
- MSP

The CLI is the key to configuring the device since it is the only way to access the configuration data base. Configuration commands may be entered one at a time or in a sequence. This later technique is used in conjunction with the Airbeam or MSP client.

4.2 *The CLI and the ConfigurationData Base*

The Configuration data base contains all the configuration information for the EWB100 device. It is stored in a special area of flash memory. Configuration changes are stored as changes from the default value. When the configuration data base is erased, the system returns to the default values for all the configuration parameters.

There are three different configuration areas in flash: two for user defined information and one for manufacturing information. Multiple changes to the same configuration value will result in prior values in the flash being invalidated and new entries being created. Hence configuration areas can “fill up” with invalidated data. Only one of the user defined areas is used at a time. When one area becomes full (usually because to too many invalidated values), all the valid data in it will be copied to the other area, which then becomes the current configuration data base and the other area erased. No such mechanism exists for the manufacturing area, since the data values are seldom altered. The state of configuration memory is shown via the “st cfg” command.

User Defined Defaults is a mechanism by which the default values of certain configuration parameters may be set by the user. These user defined defaults will override the runtime default values when the device is restored to “factory defaults.” They are limited to a few subsystems and parameters within those subsystems. The goal of this feature is to allow a device to upload new

configuration information from a remote server even if the current configuration has been corrupted. The “existing” configuration can be deleted and the “user defined defaults” will be used to access the update server.

4.3 *Accessing the CLI*

As noted above, CLI is always available on the device and can be reached from a variety of sources:

- USB
- Telnet
- HTTP
- “Command Files”
- MSP

There is a USB interface on the device that also serves at the headset jack. When used with the proper connector and with a driver on the PC, the CLI can be accessed via HyperTerm on a Windows PC. The details on how to use the USB connection is described in Appendix E

The Telnet and HTTP interfaces are available over the network.

Command Files are used to deliver a sequence of CLI command lines to the device. These are downloaded by the Airbeam or MSP client from a server. They are typically used to configure the device when there are more than just a few configuration settings or when there are many devices to be configured.

The input model is very simple. When looking for the next character input, the CLI routines examine 4 different input buffers. Each of these buffers corresponds to one of the 4 above sources. The drivers for these sources simply deposit characters into their buffer and the CLI will pull them out accordingly. With this model one can mix input from different sources, although in practice this seldom happens as only one interface is generally active at a time.

Output is directed to whatever source from which the last input character came.

The CLI syntax is simple and consists of a command followed by a number of parameters. There is a help facility that can be accessed at any time by entering a “?” (i.e. “set ?)” will display all the possible inputs to the set command). The help facility is present on most but not all commands. All commands must be entirely lower case.

Most input parameters can be shortened to the minimally unique string. Hence:

- “set” can be “se”
- “statistics can be “st”
- “key” can be “ke” or even “k”

The key to using to the shortened format is that what is entered must be unique among the possible options for the command.

4.4 *CLI Access Control*

4.4.1 Basic Concept

A password may be defined that will control access to the CLI via all mechanisms. If a password is defined, the user must enter the password when prompted (or must be included in CLI script files). If no password is defined, access to the CLI is unrestricted.

4.4.2 Configuration Commands

4.4.2.1 set Password Command

This command defines a Customer defined CLI password. If a password is defined, it must be entered before access the CLI is granted. If no password is defined, access is unrestricted. By default no password is defined. The command to set the password is:

```
set misc clipwd x
```

Where x is the password. If a password exists and the user wants to disable it, then x should be set to “none”.

4.4.2.2 Encrypted Customer CLI Password Command

This command defines the Customer defined CLI password, using an encrypted string. The string must be created using the clientencrypttool program found on Motorola’s support site. If a password is defined, it must be entered before access to the CLI is granted. If no password is defined, access is unrestricted. By default no password is defined. The command to set the password is:

```
set misc eclipwd x
```

where x is the encrypted version of the password. Note that x must be a multiple of 32 hex digits up to a maximum of 128 hex digits. If a password exists and the user wants to disable it, then x should be “none” (in encrypted form).

4.5 *Encrypted Configuration Parameters*

Certain Configuration Parameters may be sent to the device in an encrypted form that uses AES encryption. The parameters are:

- WPA Passphrase and User Default WPA Password
- WPA Key and User Default WPA key
- Airbeam Password and User Default Airbeam Password

This mechanism allows sensitive information to be placed into CLI files without a concern that the information will be compromised.

4.6 *"new", "new2", and "new3" parameters*

There is one set of configuration parameters that is somewhat different than most of the others. These are the "new" configuration items under the "misc" subsystem tag (i.e. "set misc new xx" and "show misc"). These three values (new, new2, new3) are defined as bit maps in which each bit controls a particular element of a particular subsystem. These were added as shortcuts to prevent the proliferation of many "on/off" configuration variables however they do introduce a different class of configuration variable. The general format used to set these commands is:

set misc xx yy

Where xx is "new", "new2", or "new3" and yy is a hex value containing the representation of all the bits that are to be set. If a user wants to change a bit configuration value, a "show misc" command should be executed to display the current value of the bit field. Whatever changes (set or clear) should take into account the currently set bits and not change other bit map settings. One should only change the bit fields of interest.

Certain bit fields are set by default and should be changed only when directly instructed by Motorola Service Personnel.

In the rest of this document, those configuration items that utilize the "new" fields will indicate it and will specify the "new" field and the bit location within that field that is to be set/cleared.

4.7 *Undocumented commands and parameters*

This document does not describe all commands and configuration variables that are available on the EWB100 and so users will encounter such items when using commands such as "show x" and "set ?". Many of these items are for developer or Motorola internal support use only and altering them may result in the EWB100 operating incorrectly or not at all. Hence only the commands and parameters that are defined in this document should be used or altered; executing any CLI commands not documented in this guide can cause unpredictable behavior and voids the product warranty.

4.8 *CLI Commands* ***Commands***

There are the following commands in the runtime software CLI:

```
set                  : set configuration
show                 : show configuration
stat                 : system status
system               : system control
notify               : performs a notify op
roam                 : roam History
cfg                  : configuration operations
help                 : display general user interface help
clear                : clear stats
ping                 : ip ping
version              : version of software
```

4.8.1 set

This command is used to assign values to various configuration parameters. There will always be at least 2 parameters and can be as many as 5 or 6. Parameters are grouped into categories associated with various subsystems on the device. The current set of parameters is described in this document and is associated with the subsystem description. It should be noted that there is a default value for every configuration parameter. Using the set command will override the default value and result in the value being written to flash. Changes for many of the parameters will take effect immediately while others will require a reboot. If in doubt, reset the device after changing any parameter or set of parameters.

4.8.2 show

This command is used to display the current value of the configuration settings for the various subsystems. There is generally only one parameter associated with a Show command which is the name of the subsystem. All configuration settings for a particular subsystem will be shown. It is generally not possible to display only one configuration parameter in a subsystem.

4.8.3 help

This command is used to display the syntax for the various commands. It takes no parameters.

4.8.4 clear

This command is used to clear all counters in all subsystems. It takes no parameters. It is not possible to clear the counters of only one subsystem.

4.8.5 stat

This command is used to display counters, statistics, and the contents of various dynamic tables. There is usually only one or two parameters associated with a Statistics command. Depending on the particular subsystem, the output can be a

list of counters (simple incrementing values), statistics (values that have been computed based upon various internal variables), or the contents of tables that have been built dynamically

4.8.6 notify

This command is used to play various audio phrases and to control the 3 LEDs. There are between 3 and 10 parameters associated with this command. The user can either specify a list of words from the Phrase Book that are to be played or can cause an LED to display a particular “blink” pattern. It is used primarily by developers and others who may alter the phrasebook contents.

This command is also used to alter the intensity of the LEDs.

4.8.7 cfg

This command is used to control and monitor the configuration subsystem. It takes one or two parameters that define the operation. The user can display various statistics, return the configuration to its default state, etc.

4.8.8 version

This command is used to display the software version as well as certain hardware information.

```
version
Motorola CA10 Version 1.1.1077 Realtime-CA10 No external mem used Mar 14 2012 08:52:24
EngVer = 4 HwVer = 2 nchip = noProt chip = Locosto-Lite
Hawkeye = 0x412b TI_DM = 0x5b66 ES = unknown
```

4.8.9 roam

This command takes no parameters. It displays the roam history for the EWB100 device.

4.8.10 ping

This command performs an ICMP ping to a device.

4.8.11 notes

Notes are a means of adding information to a device that is stored in configuration memory. Notes are text strings that are not used by the device in anyway but are meaningful in some way to users, support personal, or developers. Information may include customer details, past history, known issues, etc. Notes can be set and displayed. There can be up to 16 lines of notes.

5 802.11 Interface

5.1 *Basic Radio Functionality*

The following describes the 802.11 related features of the EWB 100.

- It is an 802.11 b/g compatible device. As such it operates only in the 2.4GHz band.
- It supports operation on channels 1-11 in the US and 1-14 on a worldwide basis. By default it will assume channels 1-11 are operational. For operation on other channels, contact Motorola support.
- To minimize power consumption it will scan only channels 1, 6, and 11. Other channels are ignored. This may be changed by a configuration command.
- Output power is typically 15dbm or greater.
- Receive sensitivity is typically greater than -85dbm for 802.11b packets.
- It supports OPEN, WEP, TKIP, and AES encryption.
- It supports WPA-PSK and WPA2-PSK authentication
- It supports WMM and will use it if supported by the AP. Voice traffic is sent on AC=0, data traffic is on AC=3.
- DTIM = 2 must be used for the device. Other settings will result in unreliable operation.
- The network should be configured to use as low a basic rate for beacons and broadcast packets as possible. Ideally 2 Mbits should be used but 5.5Mbits is also supported. Due to a lack of antenna diversity, the EWB100 does not operate well at 11Mbits basic rate. Pure 11g operation is supported but again the lowest acceptable rate should be used.
- To increase receive range the device will associate at lower than the highest (54Mbits) 802.11g rates, usually 24 Mbits. It will transmit at any rate supported by the AP.

- The device operates primarily in PSP mode, even during voice packet transfers. As such the EWB is very sensitive to variations in Beacon delivery times. If an infrastructure is configured to support multiple BSS, the one the EWB uses should be the first one if secondary ones experience significant variability in delivery times.

- Roaming/scanning is triggered primarily by missed consecutive beacons or by sudden and significant drops in the RSSI value. By default 5 missed consecutive beacons will trigger a scan.

- The device discovers APs only by active scanning use probe requests. It does not perform “background” or passive discovery of APs.

- The device will filter APs based on the RSSI and ignore those with RSSI below a threshold. This may be changed by a configuration command. For best performance the network should be designed to support signal strength of -70dbm or better.

- The device maintains a list of 16 APs. It refreshes this list entirely on each scan.

5.2 *Radio Configuration Commands*

This section describes the configuration commands associated with the radio.

5.2.1 ESS

This command sets the ESS for the radio. The format is:

```
set radio ess xyz
```

Where xyz is a string that defines the ess for the radio. It can be from 1 to 31 ASCII characters. This value does not take effect until the device is reset.

The default is “motorola”.

5.2.2 User Default ESS

This command sets the “user default” ESS. This value is used when the system configuration is returned to “system defaults”. It has the same syntax and default value as the ess command:

```
set radio defess xyz
```

The default is “motorola”.

5.2.3 Security modes

This command defines the acceptable security modes for the radio. The format is:

```
set radio security xyz
```

Where xyz is a string that defines the target security mode. Acceptable values are:

- none
- wep 64 (40 bit wep)
- wep128 (128 bit wep)
- tkip
- aesccmp

The default is “none”

5.2.4 User Default Security Mode

This command sets the “user default” security mode. This value is used when the system configuration is returned to “system defaults”. It has the same syntax and default values as the security mode command:

```
set radio defsecurity xyz
```

5.2.5 WEP Security Key

This command sets the key values when WEP is used. The format is:

```
set radio key x abcd
```

Where x is the WEP key index (0-3) and abcd is a WEP key value associated with the index..It is either 10 or 26 hex digits depending on the WEP option selected.

5.2.6 WEP index

This command sets which WEP key will be used for data transfers. The format is

```
set radio index x
```

Where x defines the WEP key index to be used. Its values are from 0 to 3.

The default is 0.

5.2.7 WPA Password

This command sets the WPA/WPA2 PSK Password . The format is:

```
set radio password xyz
```

Where xyz is the password string. It may range from 8 to 63 ASCII characters. This password must be set only after the ESS for the device has been specified.

This parameter cannot be displayed after being entered.

There is no default value for this parameter

5.2.8 User Default WPA Password

This command sets the “user default” wpa pasword. This value is used when the system configuration is returned to “system defaults”. It has the same syntax as the wpa password command:

```
set radio defpassword xyz
```

5.2.9 Encrypted WPA Password

This command sets the WPA password using an encrypted value. The password must have been encrypted using the cliencrypttool program with an identical key. The command format is:

```
set radio epassword xxx
```

where xxx is the encrypted password. Note that xxx must be a multiple of 32 hex digits up to a maximum of 128 hex digits.

There is no default value for this parameter.

5.2.10 Encrypted User Default WPA Password

This command sets the “user default” WPA password using an encrypted value. The password must have been encrypted using the cliencrypttool program with an identical key. The command format is:

```
set radio edefpassword xxx
```

where xxx is the encrypted password. Note that xxx must be a multiple of 32 hex digits up to a maximum of 128 hex digits.

There is no default value for this parameter.

5.2.11 WPA key

This command sets the security key when TKIP or AESCCMP are used. It contains 64 hex digits to create the 256 bit key. The command is

```
set radio wpakey xyz
```

where xyz is the security key.

There is no default value for this parameter.

5.2.12 User Default WPA key

This command sets the user default security key when TKIP or AESCCMP are used. It contains 64 hex digits to create the 256 bit key. The command is

```
set radio defwpakey xyz
```

Where xyz is the security key.

There is no default value for this parameter.

5.2.13 Encrypted WPA key

This command sets the security key when TKIP or AESCCMP are used. It contains 64 hex digits to create the 256 bit key. The WPA key must have been encrypted using the cliencrypttool program with an identical key. The command is

```
set radio ewpakey xyz
```

Where xyz is the security key.

There is no default value for this parameter.

5.2.14 Encrypted Default WPA key

This command sets the default security key when TKIP or AESCCMP are used. It contains 64 hex digits to create the 256 bit key. The WPA key must have been encrypted using the cliencrypttool program with an identical key. The command is

```
set radio edefwpakey xyz
```

Where xyz is the security key.

There is no default value for this parameter.

5.2.15 Minrssi

This command defines the minimum measured RSSI for AP to be considered acceptable. APs with a measured RSSI below this number will not be considered for association, even if they are the only APs that are visible. The command is:

```
set radio minrssi x
```

Where x is the RSSI value as a positive, decimal number. Internally the value will be converted to a negative number (i.e. -78dbm).

The default is -80dbm.

It should also be noted that when this value is displayed using the “show radio” command, it will be displayed as 65535-x where x is the value that was entered. The display routines do not handle negative numbers properly and so the value is displayed as a large positive number.

5.2.16 Deltarssi

One of the triggers for a scan is that the received RSSI drops below a certain threshold. This command specifies the threshold. The command is:

```
set radio deltarssi n
```

The default value is 10.

Where n is the RSSI threshold. Like minRssi it is specified as a positive, decimal value and is converted to a negative number internally. When displayed, it will also be displayed as large positive number.

5.2.17 Channel mask

This command defines the channels that the radio will scan on looking for APs. The format is:

```
set radio chmask xx
```

Where xx is a 16 bit hex number in which the bits correspond to the channels upon which the radio will scan Bit 1 of the mask corresponds to channel 1, bit 2 channel 2, etc... Bit 0 is not used.

The default is 0x0842 which corresponds to channels 1, 6, and 11.

5.2.18 roamMissBeacon

EWB100 uses missed beacons as one of the triggers for roaming away from the current AP. This command defines the number of consecutive missed beacons that will cause a roam to a new AP. The device will not necessarily roam to another AP but will perform a scan.

The command is:

```
set radio roamMissBeacon x
```

Where x is the number of consecutive missed beacons.

The default is 6.

5.2.19 scanMissBeacon

EWB100 uses missed beacons as one of the triggers for doing an active scan. This command defines the number of consecutive missed beacons that will trigger a scan for a new AP. The device will not necessarily roam to another AP but will perform a scan.

The command is:

```
set radio scanMissBeacon x
```

Where x is the number of consecutive missed beacons.

The default is 4.

5.2.20 roamRssi

One of the triggers for a scan is that the received RSSI drops below a certain threshold. This command specifies the threshold. The command is:

```
set radio rssiavg n
```

The default is -65dbm

Where n is the RSSI threshold. Like minRssi it is specified as a positive, decimal value and is converted to a negative number internally. When displayed, it will also be displayed as large positive number.

5.2.21 B rates

This command defines the 11b rates that will be requested as basic rates by the EWB100 when associating with an AP. The format is:

```
set radio brates xxxx
```

Where xxxx is a 16 bit hex number in which the bits correspond to the data rates that the AP will not use to send data to the device. The bits are assigned as follows:

- Bit 0 – 1 Mbit/sec
- Bit 1 – 2 Mbit/sec
- Bit 2 – 5.5Mbit/sec
- Bit 3 – 11 Mbit/sec

The default value is 0xf;

5.2.22 blockedrates

This command defines the 11g data rates that the radio will not request from an AP during the association process. The format is:

```
set radio blockedrates xxxx
```

Where xxxx is a 16 bit hex number in which the bits correspond to the data rates that the AP will not use to send data to the device. The bits are assigned as follows:

- Bit 0 – 6 Mbit/sec
- Bit 1 – 9 Mbit/sec
- Bit 2 – 12 Mbit/sec
- Bit 3 – 18 Mbit/sec
- Bit 4 – 24 Mbit/sec
- Bit 5 – 30 Mbit/sec
- Etc…

The default is 0xe0 (ie. All rates about 24 mbits are not requested during association.

It should be noted that the EWB100 can use any rate supported by the AP. This command simply limits what rates the AP can use to send to the EWB100 device.

5.2.23 Cradle low power mode

If sitting in the cradle, the device will scan for new APs less often when in the cradle than when not in the cradle. This will reduce power consumption in situations where the cradle is located in an area of poor RF coverage. The default is disabled. The function is controlled by the following configuration bit in the “misc new3” command**:**

The bit value is 0x0001.

The default value is disabled.

6 IP Stack

6.1 *Basic Functionality*

This section describes the functionality of the TCP/IP stack within the EWB100.

- The EWB supports only IPv4.
- The EWB supports three mechanisms for obtaining IP addressing information: Static, “MAC”, and DHCP. The factory default is “MAC” in which the lower 24 bits of the IP address is taken from the lower 24 bits the MAC address. The upper 8 bits of the IP address with MAC mode is 192. The means to obtain the IP addressing information may be changed by a configuration command (including all IP parameters if Static IP addressing is used).
- The device only performs a DHCP on initial startup and when the address lifetime is reached. It does not, by default, do a DHCP on a roam. This may be changed by a configuration command to use a DHCP “renewal”. This lack of a DCHP exchange on a roam may be confusing to some infrastructure products.
- The device will not generally roam across subnets since it does not perform a DHCP request sequence on roams. In such a situation the device will continue to use the IP address from the original subnet which will not be acceptable on the new subnet. Some infrastructure products permit a device to keep its IP address even when roaming onto a different subnet. The EWB will work with infrastructures so configured. This configuration may be complex so the default recommendation is to deploy in a single subnet environment.
- The device supports ICMP and has a “Ping” utililty available from the CLI
- The device supports both IGMP V2 and V3 but IGMP is disabled by default. If enabled, the interval between IGMP packets is 240 seconds but may be changed if desired.

6.2 *Configuration Commands*

These commands configure the TCP/IP stack on the EWB100 device. There are three modes for the stack: DHCP, Static, and MAC.

6.2.1 Network Mode

This command specifies how the TCP/IP stack will acquire an IP address. The format is:

```
set network mode xxx
```

Where xxx is may be dhcp, static, or mac.

If DHCP, the device will obtain IP addressing information using DHCP.

If static, then the user must enter the IP address, mask, and default gateway manually using the commands in this section.

If MAC, then the IP address is the 192.x.y.z where x.y.z are the lower 24 bits of the mac address

The default is MAC.

6.2.2 User Default Network Mode

This command sets the “user default” network mode. This value is used when the system configuration is returned to “system defaults”. It has the same syntax and defaults as the network mode command:

```
set network defmode xyz
```

6.2.3 IP address

This command specifies the IP address for the device. It is required if Static mode is selected. The format is:

```
set network ipadr a.b.c.d
```

Where a.b.c.d is the IP address for the device.

The default is a mac defined value as defined above.

6.2.4 IP mask

This command specifies the IP mask for the device. It is required if Static mode is selected. The format is:

```
set network mask a.b.c.d
```

Where a.b.c.d is the ip mask for the device.

The default is 255.0.0.0.

6.2.5 Default gateway

This command specifies the IP of the default gateway for the device. It is required if Static mode is selected. The format is:

```
set network dgw a.b.c.d
```

Where a.b.c.d is the IP address of the default gateway for the device.

The default is 192.168.0.1.

6.2.6 DNS Server IP Address

This command specifies the IP address of the DNS server for the device. It is required if Static mode is selected and DNS service is required. The format is:

```
set network dns a.b.c.d
```

```
The default is: 192.168.0.1.
```

Where a.b.c.d is the ip address of the DNS for the device. If DHCP is selected, it is supplied as part of the DHCP response and this entry will be ignored.

6.2.7 Append

This command specifies a user-supplied DNS name is to be prepended to the domain name provided by the dhcp response.

```
set network append enable/disable
```

The default value is disabled.

6.2.8 User Defined Append

This command sets the “user default” append mode. This value is used when the system configuration is returned to “system defaults”. It has the same syntax and default as the append mode command:

```
set network defappend xyz
```

6.2.9 IGMP Mode

This command enables the IGMP feature on the device. If enabled, the device will send an IGMP message on each roam and at user specified intervals. The command is:

```
set misc igmp xx
```

Where xx is off, V2, or V3 (igmp v2 or V3).

The default is off.

6.2.10 IGMP Request Interval

The command sets the IGMP packet interval. The command is:

```
set misc jgmpinterval xx
```

Where xx is the interval in seconds. Note that the command does being with a “j”, not an “l”.

The default is:240 seconds.

6.2.11 Perform DCHP on Roam

The device normally performs a DHCP handshake only on initial startup (and if configured for DHCP). If set the device will perform a DHCP “renewel” handshake on each roam. It assumes that whatever IP address was initially given will remain valid after all roams.

This feature is present to fix problems with roaming on Cisco 4400 network products that expect a DHCP on each roam. This function is controlled by the following configuration bit in the “misc new2” command:

The bit value is: 0x0080

The default is disabled.

7 Walkie Talkie

7.1 *Basic Concepts*

One of the primary features of the EWB100 is the “walkie talkie” like voice capability. A user pushes the “Push to Talk Button” and a voice stream goes out to all other devices.

This section will describe the implementation of the WTT client on the EWB. As mentioned earlier, it is compatible with the Team Express Clients that run on other Motorola devices.

There are 32 channels available which are labeled 1-32. For a given device one of those channels is considered the default configuration channel (or “home channel”) and is the channel that the voice stream will be sent upon by default when the PTT button is pressed. There are a variety of mechanisms for switching to other channels that will be discussed later in this document, such as profiles and “alternative channels”. The default configuration channel is set by a configuration command on a per profile basis.

When a user pushes the PTT button, there will be a brief delay followed by one of two tones. The first tone is a “goahead” tone that indicates the user may start speaking. The second tone is “busy” tone, which indicates some other user has already gained access to the channel, although they may not have started speaking yet. This second tone is also played if the user presses the PTT key while a voice stream is already being played.

While the device can only transmit on a single channel at any given moment, it can listen to any number of channels at the same time. In the simplest model, the device will listen only on the current home channel, but there are many cases in which the user may want to listen to multiple channels but only transmit on one of them by default. The EWB supports this mode by defining a channel map that indicates which channels it wants to listen upon. The voice streams on the selected channel will be passed through to the user. The receive channel map is set by a configuration parameter on a per profile basis.

When a user receives a voice stream on a channel other than the defined “home channel”, the device will switch the transmit channel to the incoming channel for a short, configuration defined, time. This allows a user to respond on a channel upon which they are listening, just by using just the PTT key.

At the end of a WTT session, a user will typically hear two beeps. The first is indicates either the termination of the existing WTT session or the return to home channel. The second indicates the termination of the Private Reply setup period, the operation of which is described in the next section.

7.2 *Protocol Elements*

The Walkie Talkie Protocol used on the EWB is compatible with that used in the Push-to-Talk Express products that run on EWP and MCxx products. This section describes the relevant details of that protocol.

- The voice stream is half duplex and uses packets that are approximately 450 bytes in length. The packets are sent every 200ms. This is different than most VOIP products that are full duplex and use small packets sent at short intervals. Thus the packets are tagged with AC 1 when used with WMM rather than AC 0.

- The packets are sent using multicast addresses at both the MAC and IP layers. The MAC address is :01:40:EF:40:02:02 and the IP address is: 239.192.2.2. Note that the latter is a class D address which is generally not passed across subnets by routers without explicit configuration of the routers.

- The EWB100 will communicate across subnets, provided the routers connecting those subnets are configured to pass the class D defined above. As noted previously, the EWB itself will not generally roam across subnets. This configuration may be complex so the default recommendation is to deploy in a single subnet environment.

- In the normal flow, the packets are sent by the originating device as unicast packets to the AP. Such packets are acknowledged by the AP. They are typically sent at the highest rate supported by the AP, at least on the first attempt. The EWB will rate scale on subsequent packets and over the long term to minimize lost packets and subsequent retries. The APs will rebroadcast such packets following beacons at the DTIM interval. Such packets are multicast and are not acknowledged by the receivers.

- The packet rate is 5pps (every 200ms). Given the low packet rate the load on the network from even multiple EWB voice sessions is very small.

- Different voice channels are identified only by different UDP ports in the range 5000-5031 (decimal). Channel 1 uses port 5000, Channel 2 uses port 5001, etc. The MAC and IP addresses used for the different channels are the same, only the UDP ports are different, The base port can be changed by a configuration command but will result in lack of interoperability with Push-to-Talk Express clients.

- There is a mechanism to resolve possible contention among sending stations. “Collisions” are resolved within 400ms and should not be noticed by users.

- There is an upper bound of 30 seconds on voice sessions. Sessions longer than will be automatically terminated. The maximum duration may be changed by a configuration command.

7.3 *Alternative Channels*

As noted above, normally an EWB100 is configured to transmit WT voice streams on a single channel. This channel is defined by the Profile (see below). A device is also programmed to listen on any number of channels, including the one that it normally uses to transmit, using the channel map. The default transmit channel and the monitored channels are both controlled by configuration parameters.

As also noted above, an EWB100 that receives a packet on a channel that it is monitoring (but is not the default transmit channel), will temporarily switch transmit channels so that the user can reply to the message. This window is typically 5-10 seconds (also a configuration parameter). After that time, the device will switch back to its default transmit channel or home channel.

There is often a need for a device to switch to another channel in order to communicate with a different set of users. This may be a one time event (i.e. speak to everyone in the store rather than just those in my department) or it can be a long lasting switch (perhaps I am switching departments for several hours).

This function can be done with Profiles (described below) but it can also be in a less complex manner using the SAPP or FAPP buttons. The user may switch to a different transmit channel by doing:

- a single press of the SAPP button
- a double press of the SAPP button
- a single press of the FAPP button
- a double press of the FAPP button

Each button sequence can specify a different alternative channel and audio clip. Both are set via configuration commands on a per profile basis.

This feature is simple to use, simply perform the action, and then use the PTT button to talk. The voice stream will go out on the alternative channel.

There are two modes for the alternative channel mechanism. In the first mode, the channel switch is good for one voice transmission or a configuration defined timeout (normally 5-10 seconds). The user may also switch back manually by

repeating the button action that triggered the change. After any of these events occur, the device will switch back to the default channel after the timeout period.

In the second mode, the device will stay on the alternative channel until the user requests to go back (by repeating the button action that triggered the change) or by the expiration of a configuration defined timeout. The user may make any number of WTT sessions on the alternative channel. The timeout would normally be a “long time” (ie. 10s of minutes, hours, etc.).

If the user selects a new alternative channel while on another alternative channel, the device will flip to the new alternative channel and restart all the timers. When the timers expire, it will go back to the home channel.

One can use the two channels for a variety of purposes. One could have a global channel and a department channel. One normally talks on the department channel but can always switch to the global channel if needed.

The configuration commands to enable/configure the alternative channel mechanism are defined on a per profile basis. Thus each profile can have its own alternative channel configuration.

Typical command sequences are:

set pr key sappsc n x

where n is the profile number and x is either wacs (for single operation alternative channel mode) or wace (for extended alternative channel mode).

“sappsc” indicates the profile is to be switched using a single click of the sapp button. Other choices are “sappdc” (sapp double click), “fappsc” (fapp single click), and “fappdc” (fapp double click).

set pr data sappsc n y

Where y is the channel number and the other choices are as above

set pr timeout wac n x

Where n is the profile number and x is the alternative channel timeout in seconds (for wacs one would make x small while for wace it would be long).

7.4 *Configuration Commands for WTT Operation*

While this section contains examples of CLI commands for operation of the WTT subsystem, the complete description of WTT configuration commands is in the Profiles section. This is because virtually all WTT configuration takes place on a profile basis rather than a global device basis. Hence it makes more sense to place all WTT configuration commands with profiles rather than here.

7.5 *Guidelines for Using Multiple Channels*

There are a variety of mechanisms to switch between WTT channels on the EWB100. This note describes a means to selecting which mechanism to use.

7.5.1 Zero channels (WTT disabled)

Disable WTT enable in the profile. Doing this makes the EWB100 a rather useless device in that the WTT functionality is disabled.

7.5.2 One channel

Program the pttkeydata field in user profile 0. Also set the wttrxmask parameter in the profile. The command sequence is as follows where x is the desired channel and n is the profile (normally 0)

set pr key ptt nwtt

set pr data ptt n x (x = channel number to transmit on, 1- 32)

set pr rxwttmask n z (where z = (1 << (x-1)))

7.5.3 Two channels – primary and occasionally secondary – listen on both

In this option, there is a primary wtt channel that is used most of the time, but the user occasionally wants to transmit on a secondary channel. The user wants to listen on both channels all the time.

It should be noted that if an incoming session arrives on the alternative channel, the device will automatically switch to that channel for a few seconds after the voice stream completes to allow the user to respond.

To operate this way, the “alternative channel single” mode should be used. By single clicking the sapp button (or another button/sequence as defined in the configuration), the device will switch to an alternative channel for either one WTT session or a configuration defined timeout, which ever comes first. If the user presses the sappsc (or programmed key) before either occurs, the device will go back to its primary or home channel.

The configuration for this mode is as follows, assuming sappsc (side application button, single click) is used for the channel switch and x and y are the desired channels

set pr key ptt 0 wtt

set pr data ptt 0 x

set pr key sappsc 0 wacs

set pr data sappsc y

set pr timeout wac w (w = timeout in seconds)

set pr rxwttmask 0 z (where z = (1 << (x-1)) | (1 << (y-1))

7.5.4 Two channels – equally used – listen on both

In this option, there are two channels that the user typically stays on for a long time. It could be two departments or two job related channels. The time on a channel can be minutes or hours. The user wants to listen on both channels all the time.

It should be noted that if an incoming session arrives on the alternative channel, the device will automatically switch to that channel for a few seconds after the voice stream completes to allow the user to respond.

To operate this way, the “alternative channel extended” mode should be used. By single clicking the sapp button (or another button/sequence as defined in the configuration), the device will switch to a alternative channel and stay there until either a very long timeout occurs or the user presses the key sequence again.

The configuration for this mode is as follows, assuming sappsc is used for the channel switch

set pr key ptt 0 wtt

set pr data ptt 0 x

set pr key sappsc 0 wace

set pr data sappsc y

set pr timeout wac w (w = timeout in seconds)

set pr rxwttmask 0 z (where z = (1 << (x-1)) | (1 << (y-1))

7.5.5 Two channels – listen on only one at time

In this mode the user wants to switch between two channels but only listen on the current channel. The user wants to manually switch between channels when desired.

To operate this way, two profiles should be defined, each with its own WTT channel setting and rxwttmask. When the user single presses the sapp button, the device will switch to the next channel. If the user does nothing, it will “stick” there. Fast profile switching should also be enabled.

It should be noted that the choice of which channels to listen to is defined on a per profile basis. Hence a number of options are available here. For example when operating on channel 1, the user could listen to channels 1 and 2 but when on channel 2, the user would only listen to channel 2.

The command sequence is below where x and y are the two channels

set pr key ptt 0 wtt

set pr data ptt 0 x

set pr key sappsc 0 pr

set pr rxwttmask 0 z (where z = (1 << (x-1)))

set pr key ptt 1 wtt

set pr data ptt 1 y

set pr key sappsc 1 pr

set pr rxwttmask 1 z (where z = (1 << (y-1)))

set mis new2 k where k includes the bits of the mask 0x40

7.5.6 Three or More Channels

This is basically a variation of the previous example but with more profiles defined. Up to 8 profiles/channels may be selected.

7.6 *WTT Command Reference*

As noted above, see the Profiles section for most WTT command reference. There are a few parameters that are WTT related and effect all sessions regardless of profile. Those are described in this section.

7.6.1 Private Reply Timeout

This command defines the time following the end of a WTT session, during which a private reply session can be started by a single press of the FAPP button. The format is:

```
set wtt smcto n
```

Where n is the number of seconds in which a Private Reply call can be made.

The default is 10 seconds.

7.6.2 Maximum WTT Session Duration

This command defines maximum duration of a WTT session. Sessions longer than this will be automatically terminated on the sending side. The format is:

```
set wtt sto n
```

Where n is the maximum duration of a WTT session in seconds.

The default is 60 seconds.

8 Private Reply/Unicast Voice

8.1 *Basic Concepts*

In addition to the multicast Walkie Talkie like functionality present in the EWB 100, there is also the capability to do private 1 to 1 voice conversations. This capability is called “Private Reply”. The “Push to talk” modality is kept but the voice stream is only sent between two parties. They can have a “back and forth” session using the PTT button that will remain so long as the session is maintained. When the conversation is done either party can terminate the session by pressing the FAPP button.

Private Reply sessions may be created following any WTT session. After the speaker stops talking and releases the PTT button, there is an interval in which any receipent of the voice stream may initiate a Private Reply session back to the speaker. On the EWB100, this is done by doing a quick press and release of the FAPP button. If the session is created properly, a quick succession of tones is heard by both parties. If it fails to connect a shorter tone sequence is played. If multiple users attempt to create a Private Reply sesson to the same user, only one will succeed and the other will fail and will be so informed. Which user is successful is mostly a function of network delays and cannot be predicted. Once the session is successfully created, either side can begin talking by pushing the PTT button. If both try to talk at the same time, a special alert tone is played. While the session exists, there can be any number of voice transmissions by either party. The session is terminated when either side presses and releases the FAPP again. Alternatively it will be terminated if neither side presses the PTT button for a configuration defined period (currently 20 seconds).

8.2 *Protocol Elements*

This section briefly describes the underlying network protocol elements involved in a Private Reply session. There are two protocols involved here. The first is SIP which is used to setup the session and to control who is allowed to transmit and the second is RTP which carries the voice streams.

SIP is used to handle session setup and transmission control. The SIP packets are delivered using UDP rather than TCP; retransmissions and the like are handled at the SIP level rather than by TCP. When the session is initiated, a SIP INVITE message is sent by the initiator (the device on the receiving end of the WTT session), indicating a desire to setup the session. This packet is normally followed by an exchange of two additional packets that will result in the session being created.

At that point, both sides of the session are “Idle”. When one user wishes to speak (i.e. presses the PTT button), the three-way handshake will be repeated so as to grant permission to the sender to begin sending voice packets. When the user releases the PTT button, another three way handshake will take place that returns the units to the idle state.

When either side terminates the session (or it times out), a SIP “END” packet will be sent and the session will be terminated on both sides following one or more additional packets.

The voice stream is half duplex, being controlled by the SIP handshakes. Voice packets are in a compressed format using G729 every 100ms and so contain 100 bytes of voice plus approximately 50-60 bytes of IP, UDP, and RTP headers. They are addressed to the IP address and UDP port that were defined in the SIP handshake.

8.3 *UC Command Reference*

There are no customer visible configuration values for the Private Reply capability.

9 Profiles

9.1 *Introduction to Profiles*

Profiles are a means to change a large number of configuration parameters for the device in a single step. Most of the user visible configuration parameters are set on a per-profile basis; when one switches profiles, one can change many configuration parameters at once. The rationale for profiles is there could be differences in configuration for various departments, jobs, or roles (lumber, cashier, plumbing, manager, etc) and so when a particular user selects a device, they select the proper profile for their job, role, or department.

Profiles can be used to implement something as simple as a WTT channel switching mechanism similar to the “channel” knob on a conventional walkie talkie.

Alternatively profiles can be used to replace the English language prompts with those in other languages. A given device could support prompts in up to 8 different languages.

There are 8 possible profiles with profile 0 being the default one. Users may switch between profiles (if the profile switching is enabled). One of the profiles is the default profile (normally profile 0 but any profile can be the default) and it will be loaded upon startup. If profile switching is not enabled, then the default profile will always be used. The default profile is set via a configuration parameter. The device will switch to the default profile at startup time and whenever it is placed into the charging cradle.

If profile switching is enabled, there are several ways of rotating through profiles. The first approach involves pressing a button to enter profile switching mode and then pressing the button repeatedly to rotate through enabled profiles. The device will say the audio prompt associated with the profile. The desired profile is then selected via a second button sequence. If the user does not select a profile after a period of time, the device will timeout and continue using the current profile. Typically in this mode, profile switching is entered by a single click of the SAPP with additional clicks of the SAPP used to move among the various profiles. A single click of the FAPP is typically used to “select” the current profile, if such a mode is enabled.

The second approach also involves pressing the SAPP button to enter the profile selection process and then repeatedly pressing that button to rotate through profiles, hearing the name of each profile in the process. In this approach, the device will adopt whatever the profile that was last “heard” by the user. For this mode “timeout” equals profile selection.

Which mode is used to select new profiles is defined on a per profile basis.

With either approach, the selected profile may become the “default” profile if so enabled by the configuration. If so enabled, the device will remember the last profile selected across resets. If this mode is not set, then the device will revert back to the default profile upon a reset. As noted above, this profile 0 by default.

Since there are many configuration parameters contained in a profile, it would be very burdensome if users had to configure all parameters in the profile. To address this problem the device always first loads an “enabled” profile with the values contained in profile 0. The contents of profile 0 are initially defined at compile time but can be changed at any time (other profiles will inherit the values contained in the current settings of profile 0). Once a profile has been loaded with the current values of profile 0, user defined changes to that profile will be overlaid upon the values derived from profile 0. Essentially user defined changes to a profile are a “delta” from those of profile 0. This allows users to define only the items in the profile that they wish to change from the values in profile 0.

For each profile there is an enable flag that is controlled by configuration. If a profile is not enabled, then the user may not select it, nor will it be loaded from profile 0. At least one profile must always be enabled. Each profile has an associated audio prompt that is played as the user rotates through the profiles. The audio prompt can be assigned via a configuration parameter. There is also an alphanumeric string.

Some changes to profiles may take effect immediately while others will be loaded only upon a reboot. Profiles that are changed from not enabled to enabled cannot be used until after a reboot. Trying to use such a profile prior to a reboot may cause unpredictable results.

The configuration parameters contained in profiles can be grouped into several general categories:

- Profile specific items
- General device functions
- WTT configuration
- Key/button functionality
- Timeouts for various functions
- Audio and LED prompts

Profile specific items are those that are associated with the profile itself. These parameters include:

- Enable/disable of the profile
- Audio prompt
- Text string (i.e. “name”)

General device functions enable/disable/configure various functions. These parameters include:

- WTT enable
- Private Reply enable
- Profile switching enable (defined as key function value)
- Mute enable

Parameters for WTT include:

- Channel listen mask
- Default channel (defined as key data value)

Parameters for key/button functionality include:

- The use of the following key sequences:
  - PTT down
  - SAPP single click/double click/repeat
  - FAPP single click/double click
- A data value associated with each key sequence

These parameters control more than the meaning of key sequences as they enable functions and supply data for those functions. For example:

- The data parameter to the PTT down key sequence specifies the default WTT channel
- Profile switching is enabled by assigning the “PR” function to a particular key sequence (typically SAPP single click)
- Enabling alternative channel switching by assigning either the “WACS” or “WACE” to a particular key sequence (typically SAPP single click) and the alternative channel to the data parameter associated with the key sequence.

Timeout values include, but are not limited to:

- Alternative channel timeout
- Mute/unavailable timeout
- Profile switch timeout

Configuration for audio and LED prompts allow a user to change the audio prompt for a function as well as the LED blink pattern. All of the audio alerts a user can hear can be replaced by another audio sequence. One can use this capability to establish novice/expert based prompts or change the language of the prompts. The parameters here include, but are not limited to:

- Start/end alerts for Private Reply
- Audio goahead alerts for Private Reply
- Start/end alerts for WTT calls
- Out of range alerts
- Low battery alerts
- LED blink pattern during active Private Reply calls

9.2 *Profile Initialization*

This command set the parameters associated with each profile. The parameters are divided up into groups in order to simplify explanation. The basic syntax for all the profile commands is similar but there are a few differences.

It should also be noted profile 0 is enabled by default with all of its parameters specified. When another profile is enabled, the device loads the parameters for such profiles as follows:

- Copy the current values from profile 0 into it
- Apply any profile specific configuration values to it

What this means is that for profiles other than 0, users need only define the desired differences from profile 0. It is not necessary to supply every parameter for every profile. At the same time, should a value in profile 0 be altered, it will propogate to all the other profiles unless it is explicity defined in each profile.

In addition, when setting values in a profile that were not enabled at startup time, it is necessary to restart the device in order to use the profile. Trying to use a profile that was previously not enabled without a reset will yield unpredictable results.

9.3 *General Parameters*

9.3.1.1 Profile Active

This parameter indicates that the profile is active and accessible by the user. This parameter must be explicitly set for each profile. The format is:

```
set profile active n y
```

Where n is the profile number (0-7) and y is either enable or disable.

Profile 0 always enabled by default and all others are disabled.

9.3.1.2 Name

This parameter assigns an ASCII string name to the profile. This name is not used for any purpose except when the distributed name directory function is used. It can consist of any ASCII character and must be between 1 and 16 characters. The format is:

```
set profile name n y
```

Where n is the profile number (0-7) and y is the ASCII string.

The default is the channel number.

9.3.1.3 Voice Prompt

This parameter assigns an alert to the profile. The alert is played to identify the profile to the user. The format is:

```
set profile prompt n y
```

Where n is the profile number (0-7) and y is the name of the alert.

The default is the profile number.

9.3.1.4 UC Enable

This parameter enables private reply on the device. The format is:

```
set profile uc n y
```

Where n is the profile number (0-7) and y is either enable or disable.

The default is enabled.

9.3.1.5 WTT Enable

This parameter enables WTT voice on the device. The format is:

```
set profile wt n y
```

Where n is the profile number (0-7) and y is either enable or disable

The default is enabled.

9.3.1.6 Scan Enable

This parameter allows the device to listen on multiple WTT channels at the same time. If this parameter is not enabled, the device will only listen on the channel it is set to transmit upon. . The format is:

```
set profile scan n y
```

Where n is the profile number (0-7) and y is either enable or disable.

The default is enabled.

9.3.1.7 Channel Listen Mask

This parameter defines the WTT channels that the EWB100 device listens upon. It is a 32 bit number in which the bit numbers correspond to WTT channels. The format is:

```
set profile  rxwttmask n y
```

Where n is the profile number (0-7) and y is a 32 bit hex number. Y is set according to the table:

Channel	Mask Value
1	1
2	2
3	4
4	8
5	10
6	20
7	40
8	80
9	100
10	200
11	400
12	800
13	1000
14	2000
15	4000
16	8000
17	10000
18	20000
19	40000
20	80000
21	100000
22	200000
23	400000

24	**800000**
25	**1000000**
26	**2000000**
27	**4000000**
28	**8000000**
29	**10000000**
30	**20000000**
31	**40000000**
32	**80000000**

Multiple bits may be set in the mask in order to allow the device to listen on multiple channels at once.

This function is available only is if the scan enable parameter is enabled.

The default is to listen on all 32 channels (mask = 0xffffffff).

9.3.1.8 Mute Enable

This command enables the mute function for this profile. The mute function stops audio from the speaker and is invoked by holding down the volume down button. It is used when an audio message is being received and the user does not want to listen to it. It applies only for the duration of the message. The next audio message will be played normally.

The command format is:

```
set profile mute n y
```

Where n is the profile number (0-7) and y is either enable or disable.

The default is disabled.

9.3.1.9 Unavailable Enable

This command enables the unavailable function for this profile. The unavailable function causes the device to ignore all audio messages for a specified period of time or until the user exits the command by pressing on the volume up button. It is used when a user does not want to be disturbed for an extended period of time

The format is:

```
set profile unavailable n y
```

Where n is the profile number (0-7) and y is either enable or disable.

The default is enabled.

9.4 *Key Usage*

These commands define the actions taken by various key operations for the profile. Each profile can assign different actions to the same key operation. All of the commands follow the following format.

```
Set profile key keyOp n app
```

Where *n* is the profile number and the *keyOp* can have the following values:

- sappdn - side application done
- sappsc – side application single click
- sappdc – side application double click
- fappsc – front application single click
- fappdc – front application double click
- ptt -- push to talk button

app is the application to be invoked and is one of the following:

- none – no action is assigned
- pr - enter switch profile app
- wtt - enter wtt channel (contained in data below). Only the ptt key may be assigned to this action
- wacs – switch to an alternative channel contained in the key data. Stay on that channel for one wtt session or until a timeout occurs and then return to the default channel
- wace– switch to an alternative channel contained in the key data. Stay on that channel for until a timeout occurs and then return to the default channel

By default “ptt” is set to “wtt” and “sappsc” is set to “wacs”. All other key actions are set to “none”

9.5 *Key Action Data*

This commands sets a 32 bit data field that is associated with a particular key/action combination. The format is:

```
set profile data x n z
```

Where *x* is the keyOp from the previous command,n is the profile number (0-7), and z is the 32 bit hex number. z may represent an ip address, extension, or channel, depending on the particular app value.

By default “wtt” is set to 1. All others key actions are set to 0.

9.6 *Alerts*

These commands assign entries from the phrasebook to particular profile alerts. All of the commands have the same format:

```
set profile alert x n z
```

Where x is the name of the alert and consists of one of the following:

- wtintgo - alert played at the start of an outgoing wtt session
- wtinitnogo - alert played if the outgoing wtt session fails
- wtinitend - alert played
- wtinstart - alert played at the start of an inbound wtt session
- wtinend - alert played at the end of an inbound wtt session
- scinitgo - alert played as a debugging indication when a GET request is made to the http server.
- scinitnogo -alert played when a Private Reply request is terminated by the remote end.
- scrcv -alert played at start of outgoing Private Reply call
- scend -alert played to indicate the end of a Private Reply call
- scoutstart -alert played to indicate the start of an outgoing Private Reply talk session
- scoutend -alert played to indicate the end of an outgoing Private Reply talk session
- scinstart -alert played to indicate the start of an incoming Private Reply talk session
- scinend -alert played to indicate the end of an incoming Private Reply talk session
- inring - this alert is not currently used
- outring - this alert is not currently used
- unvring - this alert is not currently used.
- hangup - alert played when full duplex call is hung up
- confirm - alert played to confirm various key requests such as status or power down
- failure - alert played when a Private Reply request is invalid
- mute - alert played when entering mute state
- unavail - alert palyed when entering unavailable state
- direnter - alert played when entering directory

- prenter - alert played when entering profiles
- swpr - alert played when the current profile is changed
- taskent - alert played when entering application state
- taskexit - alert played when exiting application state
- lowbat - alert played when battery level is low
- lostnet - alert played when lost WLAN assocation
- gotnet - alert played when WLAN association established
- wac or wacssc - alert played when going to an alt channel via single click of the SAPP
- wacsdc - alert played when going to an alt channel via double click of the SAPP
- wacsrp - sapp repeat: alert played when going to an alt channel via hold down of the SAPP
- wacfsc - alert played when going to an alt channel via single click of the FAPP
- wacfdc - alert played when going to an alt channel via double click of the FAPP
- home - alert played when returning to home wtt channel
- busy - this alert is not currently used.
- profile - alert played before each profile name
- comeup - alert played when unit powers up
- godown - alert played when unit powers down
- oonerr - this alert is no longer used.
- cerr - alert played when there is a charging error of some type
- ready - alert played when removing ready device from cradle
- notready - alert played when removing not ready device from cradle
- battery - alert that says “battery” when removing device from cradle
- l1_L1 - alert for highest battery charge state when removing device from cradle (100%)
- l2_L2 - alert for 2nd highest battery charge state (75%)…
- l3_L3 - alert for 3rd highest battery charge state (50%)

- l4_L4 - alert for 4rd highest battery charge state (25%)
- l5_L5 - alert for 3rd highest battery charge state (10%)
- l6_L6 - alert for lowest battery charge state (5%)
- nosignal - alert played for extended out of range state

n is the profile number (0-7)

z is an ASCII string that identifies an entry in the phrasebook.

9.7 *Timeouts*

These commands define the timeout values for various actions and states within the profile. They all have the same format:

```
set profile timeout x n z
```

Where *x* is one of the following and n is the profile number (0-7). *z* is the time period in seconds. The default value for each, in seconds, is shown next to each parameter enclosed in “()”.

- mute – maximum time in mute state (30)
- unavailable – maximum time in unavailable state (120)
- dir – maximum time to select an entry in the directory (5)
- profile – maximum time to select a new profile (10)
- wac – maximum time to stay on an alternative wtt channel (5)

10 User Defined Default

10.1 *Basic Concept*

User Defined Defaults allows users to alter the default values of various network, radio, and airbeam parameters. When the device is restored to “factory defaults”, these parameters will be loaded with values the user specifies rather than the factory defaults. Only those parameters needed to do an Airbeam update have user defaults. The purpose of this feature is to have sufficient configuration parameters so as to be able to reach the airbeam server even if the regular parameters have somehow been assigned invalid values.

If user default mode is disabled, then the system works identically as to previous versions. Configuration may be reset by CLI command (“cfg default”) or by holding down both the VOLUME DOWN and SAPP buttons for several seconds while the device is in the cradle.

If user default mode is enabled, the above procedures will result in most configuration parameters being restored to the factory defaults. However those with an associated user default, will have their values set to the user defaults. This includes both the CLI command and two button sequences.

10.2 *Supported Parameters*

The parameters with “user defaults” are:

- Radio ESS
- Radio Security Mode
- Radio WPA Password
- Radio WPA Key
- Radio Infrastructure type
- Network mode (i.e. DHCP/Static)
- Network “append mode”
- Airbeam enable
- Airbeam server name
- Airbeam ip address
- Airbeam port
- Airbeam package name
- Airbeam username
- Airbeam password

Syntax for these commands are described in the section for each subsystem

10.3 *Showing User Defined Default Values*

The regular and user default parameters can be shown via the regular show configuration commands :

- Show radio
- Show network
- Show airbeam

To see the user defaults, user mode must be enabled (see above). Otherwise they are hidden.

10.4 *Enabling User Defined Defaults*

User defined defaults must be enabled by a configuration command. The command is:

set misc user enable/disable

The default is disabled.

10.5 *Clearing User Default Values*

Clearing the user defaults (which have the same default values as the corresponding parameters) is done only via the command:

```
cfg default userdefaults
```

It cannot be done via any button sequence.

10.6 *Clearing and Erasing Default Values*

Clearing the user defaults and erasing the values is done via the command:

```
cfg load boot e
```

11 User Interface Customization

11.1 *Basic Concepts*

Some aspects of the user interface may be altered by configuration settings. Some define the behavior of buttons while others control how the device behaves in certain situations. This section defines what may be changed and how to do it.

11.2 *Out of range power down*

This command defines the behavior of the device if either an 802.11 association cannot be established or if IP addressing information cannot be fetched from a DHCP server (assuming DHCP mode is selected in network configuration). If the value is not zero, then the unit will power down after the indicated period of time if either of the above conditions exist. The command format is:

```
set misc oor x
```

Where *x* is the number of seconds to wait. A value of zero indicates that the unit will never power down. The value is zero.

The default value is disabled (0).

11.3 *Eliminate Flashing of Green LED*

If set, the device will not flash the Green LED as a heartbeat. The function is controlled by the following configuration bit in the “misc new” command**:**

The bit value is: 0x8000.

The default is to flash the green LED.

11.4 *Out of Range Audio Alert Repeat*

Normally the device will issue only one indication that it does not have an association. If this parameter is set, the device will issue such indications on a regular basis. The function is controlled by the following configuration bit in the “misc new2” command:

The bit value is 0x0008

The default is to repeat the out of range indication.

11.5 *Low Battery Audio Alert Repeat*

Normally the device will issue only one indication that the battery level is at critical levels. If this parameter is set, the device will issue such indications on a regular basis. The function is controlled by the following configuration bit in the “misc new2” command:

The bit value is 0x0010

The default is to repeat the low battery warning.

11.6 *Disable Clear Configuration Key Sequence*

If set, the capability to clear a configuration using a multiple key sequence is disabled. The function is controlled by the following configuration bit in the “misc new2” command:

The bit value is 0x0800

The default is to enable the sequence.

11.7 *Return to Default Profile when inserted into charging cradle*

If set, when the device is inserted into the charge cradle it will set the profile to the “default profile” (which defaults to profile 0 but can be set to any profile). The function is controlled by the following configuration bit in the “misc new2” command:

The bit value is 0x200.

The default is not to return to the default profile.

11.8 *Simplfied Profile Selection Mode*

If set and if the SAPP button is configured for Profile Switching mode, then pressing the SAPP button will rotate through the enabled profiles without any prompt other than the profile alert. Each profile is assumed to be configured for a different WTT channel and so this provides an easy way to scroll through up to 8 WTT channels (1 per profile). No confirmation is required; the the device will simply stay indefinitely on the last selected channel. The function is controlled by the following configuration bit in the “misc new2” command:

The bit value is 0x0040.

The default is to not to use the simplified mode..

11.9 *Set current profile as default profile*

If set, the device will save to flash any profile changes. The last profile selected by a user will become the “default” profile on any subsequent system restarts. The default is disabled. The function is controlled by the following configuration bit in the “misc new3” command:

The bit value is 0x0002.

The default is to disable this mode.

11.10 *Roam alert timeout*

This command defines the duration that the unit may not be associated with an AP before the “out of range” alert is played. This avoids alerts when association is lost for a short period of time.

```
set misc roamalert n
```

Where *n* is the time in seconds.

The default is 10 seconds.

12 Telnet

There is a Telnet server present inside the EWB100 that allows remote access to the CLI. It is reachable using any Telnet client. It supports one client at a time.

If the CLI requires a password, the user must enter the password before gaining access with Telnet.

The Telnet interface is intended for “lightweight” usage such as setting variables or examining configuration or statistics. It should be not be used for capturing trace logs as doing so may crash the device.

12.1 *Disable Telnet Server*

If set, the internal telnet server will not be enabled. If the telnet server is disabled and is enabled via the command, it will not startup until the device is restarted.

The function is controlled by the following configuration bit in the “misc new2” command:

The bit value is 0x0400.

The default is for the telnet server to be disabled.

13 Airbeam

13.1 *Basic Concepts*

This section explains usage of Airbeam to update and configure the EWB100

The firmware/configuration update process is used for downloading new runtime code, configuration files, and audio clips to the device. It follows the “Airbeam” model that uses FTP to download “package” files that describe new runtime/configuration information that is be downloaded. In this document it will be termed the “airbeam” process.

This section will describe the configuration/download process and associated file formats. The configuration variables that control Airbeam will also be described.

13.2 *The Configuration Process*

13.2.1 Overview

The EWB device configuration/update mechanism is based around the concept of “packages” and is roughly based on the old Airbeam model. A package is an ASCII file that contains a package number and a list of files that are to be downloaded to the device. The server is a conventional FTP server (the DA application contains an FTP server). There can be multiple package files on the FTP server, each containing a different configuration/firmware version. They are distinguished by pathname and file name. A single package file can service any number of devices if so desired. Alternatively there can be a package per device. The choice is up to the system administrator. The current DA implements a package per device model, but nothing in the model requires this.

The EWB uses FTP to access the package files. On a regular basis the EWB will log onto the FTP server (using a username and password stored on the device) and download the current package file (using the stored path and file name). It will compare the package number contained in the file with the last one it downloaded. If they are the same, nothing more will be done. If they are different, the EWB will start a process to download the files referenced in the package file.

The EWB currently does this check at various times:

- When powered on or reset
- A certain time after being inserted into the cradle
- When the user presses a certain key sequence
- When the user enters a certain CLI command

During the update process the device will go undergo multiple resets. User should wait for a “Ready” voice prompt to make sure there is no Airbeam update in progress.

The EWB also sends out a “status packet” on a user defined interval. This status packet contains lots of information such as the MAC address of the device and

network status. It also contains the latest package number stored in the device. It is sent to a particular IP address. Hence by monitoring status packets a user can determine if a device has updated to the latest package.

13.2.2 Packages

A Package is a concept taken from Airbeam and consists of a package file and one or more component files. The package file is an ASCII file, editable using Notepad or similar editors, and contains a package identifier (a number), the number of associated files, and the path to each such file. An example package file is:

packageId = 1234
componentCount = 3
Runtime : 2345 : mainflashrtload.bin
PostCfg : 6789 : postCfg.cli
Tts : 1232: english_dict.bin

The packageId is an arbitrary ascii string. Each EWB stores the current packageId and compares it against the Id in the package file. If they are different (not lessor or greater), the contents package file will be downloaded.

The componentCount is the number of component files.

The next line(s) are information about each of the component files. There are 3 fields in each line.

The first is the type of file. There are five keywords:

- Profile - profile related cli commands
- Network – network/radio related cli commands
- Postcfg - other cli commands
- Runtime (new runtime code) - binary
- Tts (text to speech files) - binary

The second is the file type version. It is separated from the file type by a colon ":". For each file type the EWB will store the latest identifier and will download the component file only if the identifier is different than the stored value.

The first three file types, if referenced in the package, is/are downloaded and executed irrespective of version. The later two files are stored as is in their respective areas in flash, only if the version is different from the one already stored in the device.

In this way it is possible to use the same package file with devices in a variety of different states. New devices will download all the component files whereas another device might only update a new binary file.

The third field is the file name. It is separated from the second field by a ":". By convention Network/Postcfg/Profile files have the "cli" suffix. Runtime/Tts has the ".bin" extension. These are by convention only.

13.2.3 Package and File Headers

The package file and each component file begins with a 128 byte header made up of a number of parameters. Each parameter is tagged with a one character identifier and all fields are separated by commas. The header is in ASCII text. An example of a header is:

I,EWB
100;Z,111;V,1829149732;B,6320080;C,b01e;N,7;S,342;D,05/04/10;T,11:10:51;

This is how to decode the header.

static	I	EWB 100
static	Z	111
dynamic	V	Version
dynamic	B	Memory location to store file
dynamic	C	CRC
dynamic	N	Flash component frame size
dynamic	S	CLI script length
dynamic	D	Date
dynamic	T	Time

Runtime files are stored at the memory location: 0x6020000.

Network/Postcfg files are stored at the memory location 0x6320000

TTS files are stored at the memory location: tbd

13.2.4 Tools to build file headers

There is an application that builds a header file (including CRC) using some input parameters and the data file. Once you have the header file, you combine it with the original data file. Here is an example

crccalc postCfg.txt 1.0 6320080 111 EWB 100 "07/18/08" "10:07:00"
cat header.dat postCfg.txt > postCfg.cli

Replace postCfg.txt with the file containing the cli commands. Replace postCfg.cli with your desired output file. You can add date and time as desired.

13.3 *Using Airbeam*

The purpose of airbeam configuration is to supply information to the device so that it knows how to obtain the package file that it is to use. The device uses FTP for this purpose and so most of the commands are FTP related. This section describes the means by which the information is supplied to the FTP client on the device.

13.3.1 Enabling the Airbeam Function

The first parameter, mode, is pretty simple. If set to enabled, Airbeam will run when any if its triggers occur. If set to disabled, it will never run. The triggers are:

- When powered on or reset
- A certain time after being inserted into the cradle
- When the user enters a certain CLI command

If enabled, the airbeam process will run when any of these triggers occur no matter how many times the trigger occurs. Upon each occurrence of a trigger, airbeam will attempt to locate the package file and act on the contents as needed.

The CLI command is: “system airbeam update”.

13.3.2 Getting the IP Address of the FTP Server

The first step in fetching the package file is to set up an FTP connection to it. In order to do so, the device must know the IP address of the server. There are five ways of learning the IP address of the FTP server:

- Specify it by setting the **ip_ab** parameter
- Specify a fully qualified DNS name via the **server** parameter
- Specify a partially qualified DNS name via **server** parameter with the remainder of the name obtained from the domain name option in the DHCP response.
- Specify an ip address in option 160 in the DHCP response.
- Specify DNS name in option 161 in the DHCP response.

The easiest way is to specify an IP address is to set the ip_ab parameter is with the command:

```
set airb ip_ab a.b.c.d
```

where a.b.c.d is the ip address of the FTP. The default value is 192.168.0.201

This approach, while simple, can cause problems if the IP address of the FTP can change or is different for every site. A more flexible approach is to specify the DNS name of the FTP via the server parameter. The command is:

set airb server ftp_dns_name

where ftp_dns_name is the DNS name of the FTP server. When the name specified the EWB will make a DNS inquiry to obtain the DNS name to IP address mapping. The IP address of the DNS server can be specified either statically in the network command or as a DHCP option. For systems that use DHCP, the latter approach is generally used and will yield both the IP address of the DNS server as well as the domain name.

The specified name can be either a fully or partially qualified DNS name. If fully qualified, the name will be submitted as specified to the DNS server. An example would be:

ftp.server.store.com

If partially qualified it will be concatenated with the local domain name and submitted to the DNS server. Hence if the server name is ftp.server and the domain name is: store123.retailer.com then the combined name would be

ftp.server.store123.retailer.com.

As noted above, the domain name is usually included as an option in the DHCP response. The option of whether to append or not is controlled by the “append” parameter in the network configuration. The command is:

set network append enable/disable

Append is disabled by default.

One can show the DNS mappings via the command: “show dns” which will list the name to IP address mappings as well as the domain name and IP address of the DNS server.

In terms of precedence, the server name takes priority over the ip_ab address as follows.

If the server name is equal to “0” (string containing the character 0) , then the ip_ab address is used.

If the server name is not equal to “0”, then the server name is used and the contents of the ip_ab parameter is not used, even if the server name does not resolve to an ip address. If the device cannot resolve the DNS name, the airbeam process is aborted.

The default server name is “0” and so by default the ip_ab address is used.

The fourth way to obtain the FTP server IP address is via DHCP option 160. If this parameter is specified in the DHCP response, the IP address contained in it

will be written to flash memory and used for subsequent FTP requests. It will also be used as the IP address for the device status packet (described below). The device will then reset itself. The update and reset will take place only if the currently defined values of the two IP addresses in flash memory is not equal to the value contained in the option. Hence once set, the value in flash will not be updated and the device will not be reset unless the contents of option 160 changes.

It should be noted that if the server name is present, the server will still take priority over the value specified in option 160 for the airbeam function.

With software version 1.1.1050, the processing of option 160, if present, is unconditional and will always take place. This may cause problems if option 160 is present for other reasons. It was made user controllable in v1052 and later releases.

The fifth method of getting the IP address is for the DNS name of the FTP server to be contained in DHCP Option 161. If present, the name contained in the option will be written to flash and used for subsequent accesses and the device reset. This update to flash and reset will take place only if the contents of the server name in flash is equal to ‘0’. This method is disabled by default but can be enabled via the command:

set misc new2mode x

Where x has the 0x1000 bit set.

In summary, the best method for setting the FTP depends on the customer site. The simplest is to set the IP address directly. More flexibility is possible using a DNS name but that requires a DNS server. If one has control over DHCP options, the DHCP options provide a way of supplying either IP address or DNS name.

13.3.3 The FTP Port

In most sites, the TCP port used by the FTP server is well known. It is defined by the FTP specification and has the value 21Hence to use a standalone FTP, the port may need to be changed, depending on how the server is configured. The command is

set airb port_ab xx

where xx is the FTP port in decimal.

The default value is 50555, which is different than the default FTP value.

13.3.4 The User Name

FTP requires a user name to login and so the device supports supplying such a name. The command is:

set airb user xx

where xx is a string that defines the user name.

If not specified and by default, the device will use its MAC address as the user name. It will be a series of 16 hex digits without separation. The DA assumes that the username will be the MAC address but the device has more flexibility.

The choice of whether to use a single username for all devices or not is a customer issue. In general, it is easiest if all devices use the same user name (and password) but it is not required.

13.3.5 The Password

FTP also requires a password to gain access to the system, even if the FTP server does not check it (such as with anonymous login). The device supports supplying the password via the command:

set airb password xx

where xx is a string that defines the user name.

If not specified (and by default) the device will use “motorola”.

It is expected that all devices that use the same username will also use the same password. The one exception to this rule would be that if the user name was “anonymous” and the FTP server was configured to accept it. In that case the password could be any string as the FTP server would not care.

It should be noted that the user may specify the password in an encrypted form. There is a separate command when specifying the password this way.

13.3.6 Package Filename

This parameter specifies the path and filename of the package file. The format of the command is:

set airb filename xx

where xx is the path and name of the package file.

If not specified (and by default) the device will use the MAC address of the device as a sequence of 16 hex digits (in which the values A-F are capital letters) with the extension “.pkg”.

There are several options when selecting the path and package name. As noted above the default is use the MAC address of the device with a null path (that is, the file is located in the home directory associated with the user name). This is how the DA program operates.

If the all or most of the devices have the same runtime and configuration values, then it maybe easier to just have all devices use the same package file. In such a case, the devices should be configured with the same filename (and path). This will greatly simplify the FTP server configuration.

If there are certain classes of device with different configurations, then each class of device may have its own filename and path.

The best option depends on the customer needs. There is sufficient flexibility to handle many situations. Since the filename is a configuration option, the value may be changed during the configuration process. Hence one name might be used for initial deployment and then changed later on. Likewise each release may specify the name of the next release.

13.4 *Airbeam and User Defaults*

Most of the airbeam configuration parameters have “User Default Values” since one of the reasons for user defaults is to able to repair a broken configuration. The user defined default commands generally follow the syntax for the associated configuration value.

13.5 *Airbeam/Device Status CLI Command Reference*

13.5.1 Mode

This command enables or disables the airbeam update process. The command is:

```
set airbeam mode <enable|disable>
```

The default is enabled. Changes take effect only after a reset.

13.5.2 User Default Mode

This command is the user defined default version of the mode comand. The command is:

```
set airbeam def_mode <enable|disable>
```

The default is disabled.

13.5.3 Airbeam FTP Server IP address

This parameter specifies the IP address of the Airbeam server. It is used onlyi if the Server name field is null. If the server name field is not null it will be used together with DNS to obtain the ip address. The command is:

```
set airbeam ab_ip a.b.c.d
```

where a.b.c.d is the ip address of the airbeam server.

The default is 192.168.0.201.

13.5.4 User Default Airbeam FTP Server IP address

This command is the user defined default version of the airbeam ip address comand. The command is:

```
set airbeam def_ab_ip xx
```

13.5.5 Airbeam FTP Port

This command defiens the TCP port used to access the airbeam FTP server.

```
set airbeam port_ab xx
```

where xx is the TCP port number. It is a 16 bit decimal number.

The default is: 50555

13.5.6 User Default Airbeam FTP Port

This command is the user defined default version of the airbeam FTP port comand. The command is:

```
set airbeam def_ab_port_ab xx
```

13.5.7 Airbeam server name

This command specifies the DNS name of the Airbeam FTP server. If specified the name will be resoled using DNS. Note that the name may either partially or fully qualified. If the network parameter “append” is enabled, then the name supplied here will be prepended the domain name provided by DHCP. If the name here is fully qualified and append is enabled, the results may not be what was expected.

```
set airbeam server xx
```

Where xx is the DNS name of the server.The default is null.

13.5.8 User Default Airbeam server name

This command is the user defined default version of the Airbeam server name comand. The command is:

```
set airbeam def_server xx
```

13.5.9 Airbeam user name

This command specifies the user name for logging into the Airbeam FTP server.

```
set airbeam user xx
```

where xx is the user name. The default is the MAC address of the device.

13.5.10 User Default Airbeam user name

This command is the user defined default version of the Airbeam user name comand. The command is:

```
set airbeam def_user xx
```

13.5.11 Airbeam user password

This command specifies the user password for logging into the Airbeam FTP server. The command is:

```
set airbeam password xx
```

where xx is the user password.

The default is “motorola”.

13.5.12 User Default Airbeam user password

This command is the user defined default version of the Airbeam user password comand. The command is:

```
set airbeam def_password xx
```

13.5.13 Airbeam Encrypted user password

This command specifies the user password for logging into the Airbeam FTP server. The password must have been encrypted using the clientencrypttool program. The command is:

```
set airbeam epassword xx
```

where xx is the encrypted user password.

The default is undefined.

13.5.14 User Default Airbeam Encrypted user password

This command is the user defined default version of the Airbeam user password comand. The password must have been encrypted using the clientencrypttool program The command is:

```
set airbeam edef_password xx
```

where xx is the encrypted default user password.

13.5.15 Airbeam filename

This command specifies the name of the package file that will be fetched by the client. The command is:

```
set airbeam filename xx
```

where xx is the name of the package file.

13.5.16 User Default Airbeam filename

This command is the user defined default version of the Airbeam package file name comand. The command is:

```
set airbeam def_filename xx
```

13.5.17 Airbeam package version

This command specifies the latest airbeam package version. It is normally not set by users as the parameter is updated by the Airbeam subsystem during the deployment process. It may be used to force the reloading of a given package however. The command is:

```
set airbeam pkg_ver xx
```

where xx is the package version.

13.5.18 Airbeam bootloader/minikernel version

This command specifies the latest minkernel package version. It is normally used and the value is updated when the minikernel is (re)loaded. Note the minikernel cannot be updated via Airbeam but must be done by CSST. The command is:

```
set airbeam bootloader_ver xx
```

13.5.19 Airbeam runtime version

This command specifies the latest runtime version. It is normally not provided by users as the parameter is updated by the Airbeam subsystem during the deployment process. It may be used to force the reloading of a given package however. The command is:

```
set airbeam runtime_ver xx
```

13.5.20 Airbeam TTS version

This command specifies the latest TTS version. It is normally not provided by users as the parameter is updated by the Airbeam subsystem during the deployment process. It may be used to force the reloading of a given package however. The command is:

```
set airbeam tts_ver xx
```

13.5.21 Airbeam Inactivity Timeout

This command specifies the maximum time that a Airbeam may wait for a response from the server for any nondata transfer request. may take. If it is not completed in this time, the process will be aborted. The command is:

```
set airbeam activity_timeout xx
```

The default is 60 seconds.

13.5.22 Airbeam Download Timeout

This command specifies the maximum time that a Airbeam download may take. This is the actual file transfer time of the data being downloaded. If it is not completed in this time, the process will be aborted. The command is:

```
set airbeam download_timeout xx
```

where xx is the time in seconds. The default is: 600.

13.5.23 Cradle Wait Time

When the device is inserted in the the cradle and the airbeam mode is enabled, it will access the Airbeam FTP serer to determine is a new package needs to be downloaded. This command specifies the delay between the insertion onto the cradle and the actual access of the airbeam server. This delay is needed to ensure that the battery has sufficient charge for the process. The command is:

```
set airbeam crdl_wait xx
```

where xx is the delay in seconds. The default is 1800 seconds.

13.5.23.1 Airbeam Package number update criteria

If set, the airbeam subsystem will require a download package number to be numerically greater than the package number last downloaded. If not set, the airbeam subsystem will only require that the package numbers be different. The default is disabled. The function is controlled by the following configuration bit in the “misc new2” command:

The bit value is 0x0004.

The default is disabled.

13.5.23.2 Load Airbeam Server Name from DHCP Option 161

If set, the device will obtain the Airbeam server name from DHCP option 161, if present. In addition the server name in the airbeam configuration parameter must also be null. It is disabled by default. The function is controlled by the following configuration bit in the “misc new3” command:

The bit value is 0x1000.

The default is disabled.

13.5.23.3 Load Airbeam Server IP from DHCP Option 161

If set, the device will obtain the Airbeam server IP address from DHCP option 160, if present. It is disabled by default. The function is controlled by the following configuration bit in the “misc new2” command:

The bit value is 0x2000.

The default is disabled.

14 MSP Support

The EWB100 supports MSP for configuration and firmware updates. Information MSP and how to use it to support the EWB100 is found in the MSP User Documentation. This section describes configuration commands for MSP

There are two MSP configuration commands and they only allow MSP to be disabled. MSP can only be enabled via the MSP staging process.

14.1.1 MSP Mode

This command disables the MSP client. The format is:

```
Set msp mode disable
```

The default is enabled.

14.1.2 MSP Default Mode

This command disables user default mode of the MSP client. The format is:

```
Set msp def_mode disable
```

The default is disabled.

15 Misc CLI Commands

This section describes a number of CLI commands that are “standalone” and not associated with another subsystem.

15.1 *Ping*

This command performs an ICMP ping.

The format is:

Ping [-x] [-t] [-c n] [-l m] [a.b.c.d]

- a.b.c.d is the ip address of the target
- -t indicates the ping is to be run continuously until stopped
- -c n indicates that n pings are to be sent. If neither –t nor –c is specified, one ping will be performed.
- -l m indicates the length of the packet.
- -x indicates the ping is to be stopped. This is used instead of a Ctl-C.

a.b.c.d is the IP address of the destination. It is not required for the –x option

There is an output displayed for each ping generated as well as summary when the ping is ends

```
Response In 24 ms
Response In 49 ms
Packets: Sent = 183, Received = 183, Lost = 0
```

If there is not response, the output will be “timed out”

It should be noted that the first ping command will fail unless the MAC address of the target is known. This is due to the limitations of the Kwiknet ARP implementation.

15.2 *The Clear Command*

This command clears many of the counters associated with all subsystems. It takes no parameters. The command is simply:

clear

15.3 *The Cfg Command*

This command modifies the state of the configuration subsystem. There are two possible parameters for this command:

- default – clears the configuration database which effectively restores the device configuration to its default value. All profiles and directory entries are deleted.
    - If user defaults are enabled, this command will set the configuration parameters to the user defaults.
    - If the phrase “userdefaults” is added to this command, the user defaults will also be cleared.
- swap – Switches between the two areas in flash that store non-manufacturing related information
- resetclear – clears the configuration database and resets the device immediately.

15.4 *Notes*

This command sets the notes entries in the configuration data base. The command is:

set notes x y

Where x is the note number and ranges from 0 to 15.

y is a text string that is associated with note. It can be up to 64 characters. If it omitted then the current note associated with the number is deleted.

15.5 *Reset Device*

This command causes the EWB100 to do a reset and restart. The format is:

```
sys reset
```

15.6 *Power Off Device*

This command causes the EWB100 to power down. The format is:

```
sys off
```

Before powering down, the device will display a message to give the user a chance to unplug the USB cable. If not unplugged, the device will immediately power back on. The message is:

remove device

15.7 *The Notify Command*

The notify command is used to invoke various notifications and alerts. Its use is primarily for testing. There are several different formats of the command.

15.7.1 For invoking notifications

The command format is:

```
notify l led pattern
```

Where *led* is an led and can have the value:

- green
- red
- blue

Pattern defines the blink pattern used and can have the value:

- on – always on
- off – totally off
- blink – one short blink
- twice – two short blinks
- slow – slow blink pattern
- fast – flast blink pattern

Each led can be given its own pattern. Fast and slow patterns keep running until set to a different pattern.

15.7.2 For invoking audio alerts

The command format is:

```
notify w p1 p2 p3 .. pn
```

Where p1, p2,… are ascii strings that identify the alerts in the phrasebook that are to be played. Each alert is played only once.

16 Audio Services

16.1 *Introduction*

This document describes how EWB100 creates audio notifications (i.e. “alerts”) for the user. It will explain the underlying mechanisms as well as how the notifications can be altered. It will also explain how the mechanisms are used to provide Multilanguage support.

16.2 *Audio Clips and Audio Tables*

The EWB100 device does not have a display and only very limited LEDs and so generally uses audio alerts to communicate information and status to the user. The audio alerts can be tones or spoken words or any combination of them. EWB100 does not have a TTS engine and instead relies on combining prerecorded audio clips to create the alerts.

Each audio clip has the following properties:

- A name which is a case sensitive ascii string
- The audio format (g729, g711u, g711a, etc.)
- The length in bytes
- Audio data
- An index into one of a number of tables (termed “clip tables”) each of which contain one or more audio clips

The name of the clip is used to identify it during configuration. It is derived from the WAV file used to build the clip. Names must be unique within a clip table but not between clip tables.

The audio format indicates which compression is used for the clip. Voice clips generally use G729. Tones use G711u or G711a (usually 711u, 711a is not used for any clips right now). G729 clips consume 1KB/sec. G711 consumes 8KB/sec. Audio clips are organized and stored as clip tables. Any given clip table stores clips that have the same audio format. Hence there are g729 clip tables, g711u clip tables, g711a clip tables, and pcm clip tables. Each clip table contains a directory of the clips stored in the table. The directory is simply a list of the clip names. Associated with each directory entry is the length of the clip and an offset in the table to the beginning of the clip. The entries in the directory are stored in a manner as to support a binary search for performance. Clip tables are internally numbered 0 to N.

An audio clip is uniquely identified by its clip table number and an index into that table. Clips with the same name can be distinguished by the combination of clip table and index. This combination can be termed the “clip-id”.

There are approximately 400 audio clips stored in the runtime image. This list of clips is called the Default Directory and includes: letters, numbers, status and configuration phrases, common phrases, various tone sequences, phrases for demo applications, etc. The Default Dictionary is basically a list of clip names. All the audio clips that the device needs to function are defined in the Default Dictionary. The contents of the Default Dictionary are listed in Appendix E of this document.

There are 3 sets of clip tables:

- Default
- Flash downloaded
- RAM downloaded

The Default set of clip tables are stored as part of the basic runtime software and contain all the clips defined in the Default Dictionary. Most are g729 clips but there are few 711 and PCM clips.

The second set of clip tables can be stored in a dedicated flash area of memory. There is currently about 1.75MB of flash memory reserved for downloaded audio clips. These clip tables are the primary means by which clips are added to the device by customers.

The third set of clip tables is stored in RAM. These would be used to store very dynamic clips that would be altered frequently. The player routines handle such tables but at present there is no way to load them into RAM.

There can thus be up to 12 clip tables in a system, numbered 0 to 11. Tables 0-3 are the default tables, 4-7 are the flash downloadable tables, and 9-11 would be the RAM tables.

Whenever the device wants to play an audio sequence, it will build a list of one or more clip-ids and submit it to an internal routine that will play out the sequence. Clip-ids are used in a variety of data structures throughout the system.

16.3 *Clip names and clip-ids*

Clip-ids are never seen by any user, including those doing configuration. They are an internal data structure. What users see during the configuration process are the clip names. The internal audio software maps from the clip name to a clip-id by searching the clip tables. The clip tables are searched in the following sequence:

RAM Tables => Flash Tables => Default Tables

And

G729 Tables => G711u Tables => G711a Tables => PCM Tables

With this model, users can select a more dynamic form of the clip in place of a less dynamic form. Hence any clip in the Default tables can be overridden by a clip in the Flash tables. Likewise any clip in the Flash tables can be overridden by a clip in the RAM tables.

The search of the clip tables to build clip-id is done at two different times. The first time is at startup when the device builds all the clip-id based on the Default Dictionary. For each name in the Default Dictionary, the device will search the available clip tables in order outlined above and build the clip-id. Hence any of the Default Dictionary clips can be overlaid by either Flash or Ram tables.

The second time the search is done is when a data structure that uses an audio clip is referenced for the first time. These are mostly likely clips specified during the configuration process, such as the name of a profile or one of the audio alerts associated with a profile.

16.4 *Clip Names, User Notifications, and Configuration*

On EWB100 devices, the audio clip associated with any given audio alert can be changed using Clip names and/or downloadable Clip Tables.

Audio messages to the user fall into one of three categories:

- Device wide
- Per Profile
- Directory Entries

Device wide audio messages use audio clips defined in the Default Dictionary. They are mostly used for status and configuration messages. To change them requires overlaying the clips defined in the Default Dictionary. These are targeted mostly at support personal (ie. “IP address”) but some can be heard by regular users as well (such as the volume up/down tone). The Default Dictionary is contained in Appendix E.

Changing the audio clip defined in the default dictionary can only be done by placing the name of the audio clip in either a flash or ram clip table along with the new audio clip. When the system starts up, as described previously, the system will use the first clip name found in the clip table and build the clip-id referring to it rather than the default table.

Per profile messages are most of the audio prompts that are visible to everyday users. These include alerts that indicate the start or end of a voice session, battery status, radio connectivity, etc. There are default values for all of these prompts that use the entries in the Default Dictionary. The defaults can be overridden by configuring the profile entry with new clip name. They can also be changed by redefining the clip associated with the entry in the default dictionary.

Changing the default value in the profile is the easiest and can be done by the command:

set pr alert xn z

Where x is the name of the alert (lowbat, outofrange, etc), n is the profile number (0-7), and z is the new clip name. The clip name must be defined in some clip table. Otherwise the silence clip is used by default.

Changing the meaning of a default clip is the same as redefining a device wide clip and uses the same technique as describe above.

Directory Entries are the names of users or channels that are defined in the directory. Typically such entries would be defined in the Flash or Ram tables, although a user might use clips in the default dictionary (there are some clips in the default dictionary for exactly this purpose). The directory is empty by default. Entries to the directory are global across all profiles.

16.5 *Multilanguage support*

16.5.1 Basic Model

The EWB100 audio processing model has been designed from the beginning to support multiple languages concurrently on the same device. This section outlines how multiple languages could be supported now and what changes could be added if the need arises.

As described above, every single audio clip defined in the default directory can be replaced by another clip in another language. All that would be required would be to define a set of click tables that contain audio clips with the same name as those in the default table. These would be placed in Flash and would completely replace the clips in the default tables. Alternatively one could selectively replace only those clips that were actually used (since currently much of the default dictionary is not used).

Replacing the default dictionary would allow the creation of a new and single language version of the EWB100 device. All of the English clips would be replaced by French or German or Spanish clips.

For the profile defined clips, the previously defined model works fine. One would define new language clips and give them unique names. These names would be assigned to profiles via configuration commands. The corresponding clip tables would be stored in Flash (or RAM). When a profile was access that contained one of these new clips, the system would do a lookup and build the correct clip-id and use it going forward. One could have a French profile and a Spanish profile. Each profile would use a unique set of clip names. The Flash clip table would contain both sets of clips. Again all this is currently supported.

If one wanted to achieve multiple language support by combining the two languages on phrase by phrase basis, one could also achieve this by creating the dual language clips, assigning them unique names, and downloading a clip table containing them. All this works using the current profile/audio clip model.

Directory entries would be global and so multiple language support would be less robust. One could either go with global clips (people’s names are generally language independent) or go with a model in which the different forms of the names are in a single clip (this would work for department names). Again, all this works in the current implementation.

There are a couple of limitations to the current implementation model.

The first is global nature of the audio clips used for status and configuration purposes. One can overlay them into any single language one wants but multiple language support is not present. For example, the phrase “IP address 10.1.2.3” could be spoken in English or French but not both. Given that the most common use is for support, this may not be an issue.

The second issue is that the above model assumes that different language support can be achieved by simple word or phrase replacement. Differences due to word ordering are not supported. For example, numbers such as “119” are parsed as “one hundred”, “nineteen”. In the default dictionary there are clips for “one”, “hundred” and “nineteen” which works in English. Whether it works for all languages is unknown. Likewise, the value “23” (which in English is “twenty” and “three”) in some other language has a unique phrase (rather than combining two separate phrases) this could be a problem for the device. Changing this could be a major task.

16.5.2 Detailed Instructions

This section describes how to configure the EWB100 to use non-English voice prompts when not using MSP. This may be done via the Airbeam package files with the DA or via the FTP transfer mechanism. It may also be done line by line using the CLI but this approach may take a long time if many devices are configured.

Audio prompts are set via normal CLI configuration commands like all other EWB100 configuration items. For each audio prompt there is a corresponding audio clip identifier. One can assign any audio clip identifier to any audio prompt. There are approximately 30 audio prompts required for normal operation of the EWB100. Additional prompts may be required if other features beyond PTT and Private Response are used.

The required CLI commands are shown below:

```
set pr alert prompt 0 xx_default
set pr alert directory 0 xx_dictionary
set pr alert failure 0 xx_failure
set pr alert mute 0 xx_n_mute
set pr alert unavail 0 xx_n_unavailabe
set pr alert direnter 0 xx_dir_enter
set pr alert prenter 0 xx_profile_enter
set pr alert swpr 0 xx_n_profile_switch
set pr alert taskent 0 xx_n_task_enter
set pr alert taskexit 0 xx_n_task_exit
set pr alert lowbat 0 xx_n_low_battery
```

```
set pr alert lostnet 0 xx_n_lost_network
set pr alert gotnet 0 xx_n_got_network
set pr alert nosignal 0 xx_nosignal
set pr alert wac 0 xx_everyone
set pr alert home 0 xx_n_home_channel
set pr alert busy 0 xx_n_busy
set pr alert comeup 0 xx_n_comeup
set pr alert godown 0 xx_n_go_down
set pr alert conerr 0 xx_n_cerr
set pr alert ready 0 xx_ready
set pr alert notready 0 xx_notready
set pr alert battery 0 xx_battery
set pr alert l1 0 xx_100pc
set pr alert l2 0 xx_75pc
set pr alert l3 0 xx_50pc
set pr alert l4 0 xx_25pc
set pr alert l5 0 xx_low
set pr alert l6 0 xx_critical
```

where xx may be:

fr - French

du - Dutch

gr - German

po - Portuguese

sp – Spanish

blank (without both the “xx” and the “_” - English

Thus to alter the language prompts simply add the above lines to the configuration file and replace the “xx” in each line with that of the desired language.

It should be noted that prompts are defined on a per profile basis and so one can set each profile to be a different language. At the same time if one wants the same language prompts to be across all profiles, one need only assign the language prompts to profile 0 and all other profiles will “inherit” the language prompts as defined above.

These additional prompts can go into the “special commands” function of the DA or in the regular configuration files if just using FTP with Airbeam packages.

In addition to the configuration values the additional language prompts must have been loaded into the flash memory of the EWB100. At this time there is only one such file required and it supports all of the above languages. The file name is: euro_tts_flash_image.bin. It may be loaded as part of the Airbeam Package file mechanism that is described in the EWB100 User Guide. This file can be found in the EWB100 section of http://support.symbol.com.

16.6 *Building New Audio Clip Files*

Please contact your Motorola Support Engineer on the mechanism for doing this.

17 Web Server

17.1 *Basic Concepts*

EWB 100 supports creation of HTTP server connections. Each server connection has to have a body of code behind it to process incoming connections. Although EWB 100 technically includes a “web server”, it is really a very simple dispatching of requests to dedicated subroutines which will process the HTTP request payload and send a response. Although the system will be extensible and can handle multiple parallel requests, the limited nature of the work to be done leads to a minimal implementation sitting atop the HTTP server protocol functionality.

There is a dispatch functionality sitting above HTTP server protocol functionality. The server dispatch functionality is really just a special callback which is attached to an HTTP server socket. It performs all the callback functions associated with incoming HTTP requests it is expected that server dispatch development will generate functionalities for various functions that we expect to support.

The following functionality can be invoked via the HTTP server:

1. Playing Audio
2. LED Control
3. System Reset
4. CLI Command Execution

17.2 *Detailed Description of Functions*

17.2.1 Playing Audio

Playing audio requires one URL parameter to specify the audio to be played. The example below calls out the phrase “Manager to Register 5” to be played, but it could just as easily call out an audio tone.

```
http://192.168.0.104/audio?play=Manager+to+register+5
```

17.2.2 LED Control

Controlling LED takes in two arguments they are color and pattern. Color can have following values.

1. Green
2. Red
3. Orange

Pattern can have following values.

1. off
2. once
3. slow
4. fast
5. on
6. twice

Example to turn LED green and blink

```
http://< device addr>/led?color=green&pattern=slow
```

17.2.3 System Reset

System reset command will reset the system. It does not take any parameters:

```
http://192.168.0.104/reset
```

17.2.4 CLI Command Execution

Any CLI command can be sent to a device and executed and the command output is sent back to the requestor in the response. There is a single “cmd” parameter which specifies the command to be executed. If the command contains spaces, then they are replaced with plus signs (“+”) as is standard for URL encoding. For example, the URL:

```
http://192.168.0.104/clicmd?cmd=ver
```

Causes the following plain/text response:

```
ver

Motorola CA10 Version 1.1.920 Realtime-CA10 No external mem used
Oct  1 2009 13:20:06
EngVer = 4 HwVer = 2 nchip = noProt chip = Locosto-Lite
Hawkeye = 0x412b TI_DM = 0x5b66 ES = unknown
```

17.3 *Configuration Commands*

17.3.1 Disable HTTP Server

If set, the device will disable CLI via the HTTP server. Other HTTP functions will still operate. The function is controlled by the following configuration bit in the “misc new3” command**:**

The bit value is 0x8000.

The server is enabled by default.

17.3.2 Port

This comment specifies the TCP port number used by the http server. The command is:

```
set http port xx
```

where xx is the TCP port used by the http server.

The default value is 80 (decimal).

18 Obtaining Device Status Information

18.1 *Basic Concepts*

The EWB100 does not have a graphical interface and so the user cannot look at a screen in order to determine the state of the device. However there are four mechanisms by which a user can determine the state of the EWB100:

- LED Blink patterns
- Audio information invoked by holding down down keys
- A Status Packet that is sent on a regular basis
- The Statistics commands in the CLI
- Roam Command

18.2 *LED Blink Patterns*

The Red and Green LEDs indicate the general state of the device. If the Green LED is blinking on a regular basis (every 10 seconds or so), it means the device is associated and authenticated with the 802.11 network and that it has an IP address.In general, all is well.

If the Red LED is blinking, then there is a problem of some kind. The number of Red LED flashes indicates the problem.

- One Flash Low battery
- Two Flashes Not associated or low signal strength
- Three Flashs Cannot obtain IP Address using DHCP
- Four Flashes Associated but cannot authenticate

18.3 *Status Key Sequence*

18.3.1 Usage

If one holds down the WTT and SAPP buttons, the EWB100 will play audio clips that contain status/configuration information. As one holds the keys down one will hear a series of beeps separated by several seconds. The information that is played a function of how many beeps are played before the keys are released.

18.3.2 Information Available

After one beep basic status information is played. This includes:

- Association status
- Signal strength in dbm
- Battery strength
- Current profile

After two beeps important configuration is played. This includes:

- IP address
- IP mask
- IP default gateway
- Lower 16 bits of current BSS (in hex)
- MAC address of device (in hex)
- Current radio channel
- Current ESS (spelled character by character)
- Current 802.11 security (wep, aes, etc)
- Software version

After three beeps a few counters/statistics are played. This includes:

- The number of roams
- The time since the last roam
- The lower 16 bits of the previous BSS (in hex)
- The percentage of transmit retries
- The percentage of missed beacons
- Battery temperature

The exact information played out may be changed from release to release.

18.3.3 Configuration Command

This command controls what information is played when the PTTand SAPP buttons are held down to trigger the “status message”. The format is:

```
set misc status n
```

Where *n* is a value between 0 and 3. The meaning is:

- 0 = disables the status message
- 1 = enables playing dynamic information
- 2 = enables playing dynamic and configuration information
- 3 = enables all information

18.4 *Status Message*

18.4.1 Device Status Update Mechanism

The Device status update process involves the device transmitting via UDP a collection of configuration and status/history information to a particular IP address. The contents of the packet are ASCII strings contains the name and value of each item. It includes items such as:

- MAC address,
- Software version
- Battery state and charging history
- Indication of whether the device is on the charger or not
- The number of WTT sessions
- 802.11 association and roaming information
- Other misc information

It is currently about 1000 bytes. It is transmitted on a regular basis by the device with the default being every 1 minute.

The current contents of the status packet are listed in Appendix A.

The Device Status function is termed “deployment” in the CLI configuration commands. The reason for this is historical as the initial use of the Device Status function was to drive the Deployment Application. Since then it has been enhanced to carry all types of status information and that is now its primary use. However the “deployment” term remains. In this document, the phrase “device status” will be used except for the actual CLI commands.

The Device Status (or Deployment) function has its own set of configuration variables that are entirely separate from those of airbeam. The fact that they are both under the “arib” systems may be confusing. Again this is due to the initial tie-in of these two functions in the DA program but they are in entirely independent of one another.

As noted above, the device status function is quite simple, it simply sends, at a specified interval, an UDP packet containing configuration and status information to a particular UDP Port on a particular IP address . Hence the configuration must specify:

- Whether the function is enabled or not
- The IP address where the packet is to be sent
- The UDP port the system with the IP address
- The sending interval

18.4.2 Enabling sending the Device Status Packet

The device status packet is always sent once after a reset. This again is a legacy of the DA program.

Whether it is sent again after that depends on the configuration command:

set airb deployrecord xx

where xx can be “manual” or “auto”. “auto” enables the sending of the device status packet on a regular basis. “manual” enables the sending the device status packet only after the user presses a particular key sequence. Again this is a legacy of the DA model.

The default is “auto”.

18.4.3 Specifying where the packet is to be sent

As noted above, the device status packet is sent to a UDP port on a system with a particular IP address. The commands here are simple. The IP address is set with the command:

set airb ip_deployment a.b.c.d

where a.b.c.d is the ip address of the system

The default is 192.168.0.201, the same as for airbeam.

The UDP port is set via the command:

set airb port_deployment xx

where xx is the UDP port as a decimal number.

The default is 48888 which was used on the DA.

18.4.4 Specifying how often the packet is to be sent

The device status packet is sent on a regular basis. This interval is set via the command:

set airb udp_period xx

where xx is interval between packets in seconds.

The default value is 60 seconds. For most systems, this is far too often. The value was set for engineering test purposes and for the DA program. For systems that are neither in a diagnostic state nor using the DA, a larger value is more reasonable. Values of 15 minutes (900 seconds) to an hour (3600 seconds) are entirely reasonable.

18.5 *Roam Command*

The roam command displaces the last 16 roams by the 802.11 radio. It is a circular buffer that displays the time, access point, and reason for each roam. It takes no parameters. The output format is:

```
--time: 60.355


   0: assoc: 11618: 00A0F8A1D589

   1: assoc: 11667: 00A0F8A1D589

   2: link dn: 44407, reason: lost ap

   3: assoc: 44900: 00A0F8A1D589

   4: assoc: 44905: 00A0F8A1D589
```

Each line has the following fields:

- Sequence number that increases with each event.
- Action (assoc indicates association, link dn indicates loss of assoc)
- Time of event from system start
- AP MAC Address or reason for loss of association

One may often have two events that are the same association event, separated briefly in time.

18.6 *The Statistics Command*

This command displays various counters and dynamic tables associated with the various subsystems on EWB100 devices. For most subsystems there is only one parameter associated with the command that is the name of the subsystem. A few subsystems may have a second parameter that identifies a particular set of items within the subystem.

18.6.1 Wireless Driver Counters

This command lists counters associated with the WLAN driver. Many of these counters are meaningful only to developers who are familiar with internals of the WLAN driver. The few that may be of more general interest are noted below (‘*’). There are two forms with and without a parameter (any value will do). More counters are shown with the parameter

First Form

```
>st wd

--time: 105.704

icm in                    : 107
controlIn                 : 36
trapsIn                   : 6
beaconsIn                 : 6
framesIn                  : 71
ucIn *                    : 4
mcbcIn *                  : 67
packets in                : 1
icm out                   : 71
ic out busy               : 0
packets out *             : 14
isr int                   : 108
nt int                    : 108
timer int                 : 60
gp2 !=0                   : 107
sleep                     : 50
```

Second Form

```
st wd 1

--time: 114.355
icm in                    : 111
controlIn                 : 36
trapsIn                   : 6
beaconsIn                 : 6
framesIn                  : 75
ucIn                      : 4
mcbcIn                    : 71
packets in                : 1
icm out                   : 71
ic out busy               : 0
packets out               : 14
isr int                   : 112
nt int                    : 112
timer int                 : 60
gp2 !=0                   : 111
sleep                     : 54
gp2=0 int                 : 0
```

```
bad gp2                   : 0
no rx bf                  : 0
tx err                    : 0
ic tx err                 : 0
tx dropped                : 0
pkt out err               : 0
trap ovr                  : 0
no kn buf                 : 0
no vo buf                 : 0
not nt q1                 : 2
not nt qn                 : 0
```

18.6.2 UMAC Counters

This command lists some of the UMAC statistics. There are two forms, with and without a qualifier. The options are shown below:

18.6.2.1 No Parameter

These counters record the total unicast, multicast, and retry packets.

```
>st umac

--time: 121.603

tx okay                   : 30
tx mc                     : 0
tx 1 rt                   : 0
tx >1 rt                  : 0
```

18.6.2.2 Receive packet counters

These counters record the numbers of different types of nonvoice packets received. Voice packets are counted elsewhere.

```
>st umac rx

--time: 125.315

rx                        : 79
rx uc                     : 1
rx mc                     : 0
rx assoc                  : 1
rx auth                   : 1
rx bcn                    : 69
rx pr                     : 1
rx dtim                   : 52
rx disa                   : 0
rx deauth                 : 0
rx dup                    : 0
rx icv err                : 0
```

18.6.3 Transmit packet counters

These counters record the number of each different type of packet transmitted. Voice packets are included in these counts. It also indicates the current transmit rate (in 0.5Mbit increments) and missed beacon %.

```
>st umac tx

--time: 128.120

tx rate                  : 22
miss bcn %               : 0
tx failure               : 0
tx null                  : 14
tx null fl               : 0
tx poll                  : 0
tx assoc                 : 1
tx auth                  : 1
tx mc                    : 0
tx uc                    : 30
tx retry                 : 0
```

18.6.4 Roaming statistics

These values record a number of items related to roaming including the total number of roams, the missed beacon %, rssi values, etc.

```
>st umac roam
--time: 131.968

roams                    : 0
missed bcn               : 0
scan rsn                 : 0
old rating               : 0
new rating               : 0
```

18.6.5 Scanning counters

These counters record the number of full and partial scans performed.

```
>st umac scan

--time: 135.049

full scans               : 1
prt scan dn              : 0
```

18.6.6 Receive Rate Histogram

These counters show how many packets have been received at each rate. Only nonvoice packets are included.

```
>st umac rrx

--time: 155.728


1:   00
2:   00
5.5: 00
11:  00
6:   00
9:   00
12:  00
18:  00
24:  00
30:  00
48:  00
54:  00
```

18.6.7 Transmit Rate Histogram

These counters show how many packets have been transmited at each rate. All packet types are included.

```
>st umac rtx

--time: 155.728

1:   00
2:   00
5.5: 00
11:  00
6:   00
9:   00
12:  00
18:  00
24:  00
30:  00
48:  00
54:  00
```

18.6.8 Transmit Retry Histogram

These counters show how the distribution of packet transmit retries

```
>st umac rtx

--time: 155.728

0:   00
1:   00
2:   00
3:   00
4:   00
…
15:  00
```

18.6.9 AP Tables

This command displays information about the current and known access points. There are two forms of the command, with and without a parameter. The first form gives information about the currently associated AP, the second gives information about all known APs.

First Form

```
>st ap

--time: 147.882

SysTime:  147891

Macadr    :  00A0F8A1D589
Chan      :  11
Aid       :  1
CapInfo   :  00000000
Rssi      :  36
Assoc Time:  11794
```

Second Form

```
>st ap a

--time: 150.007

*00A0F8A1D589 -c11 -q36 -c0021 -r000F:0003 -a0
…. All known aps listed.
```

- First c is channel
- q is rssi from scan
- Second c is capabililities
- -r a:b where a are supported rates and b are basic rates
- -a is the association time

18.6.10 Receive Packet Statistics

This command displays information received packets. It shows informationon the last 16 received packets. Information includes:

- Time in microseconds since last packet
- Received data rate
- RSSI
- Packet size

The table wraps around on itself.

```
>st rx

--time: 152.707

00014657:    0   37  110
00022457:    0   36  110
00030258:    0   36  110
00038057:    0   39  110
00041957:    0   37  110
00043907:    0   37  110
00047807:    0   39  110
….
```

18.6.11 Voice Packet Transmit Statistics

This command lists the number of retries for the last 32 transmitted voice packets. It is a circular list wraps on itself. It also indicates the total transmits and average packet retry count over the 32 packets.

```
>st vtx

--time: 166.556


00: 00
01: 00
02: 00
….
31: 00

txCnt: 0, avg rt: 0
```

18.6.12 TCP Counters

This command provides information about the number of TCP connections that have been attempted and established as well as the number and types of packets that have been transmitted and received.

```
>st tcp

--time: 179.689

ConAtt                    : 1
ConAcc                    : 0
ConEst                    : 0
ConDrop                   : 0
ConDrTo                   : 1
ConDrKe                   : 0
ConRtTo                   : 12
ConPeTo                   : 0
SndPkTot                  : 13
SndPkDat                  : 0
SndPkRt                   : 0
SndPkAck                  : 0
RcvPkTot                  : 0
RcvPkDat                  : 0
RcvPkDup                  : 0
RcvPkOor                  : 0

RcvPkDuA                  : 0
RcvPkAck                  : 0
```

18.6.13 UDP Counters

This command provides information about the number of UDP packets that have been transmitted and received.

```
>st udp

--time: 182.507

 In                       : 0
 NoPort                   : 0
 InErrs                   : 0
 Out                      : 3
```

18.6.14 ICMP Counters

This command provides information about the number of ICMP packets that have been transmitted and received.

```
>st icmp

--time: 184.940

InMsg                    : 0
InErr                    : 0
InEcho                   : 0
InEchoR                  : 0
OutMsg                   : 0
OutEcho                  : 0
OutEchoR                 : 0
```

18.6.15 Arp Tables

This command displays information on both the number and types of ARP packets that have been transmitted and received as well as the current “arp table” that provides IP address to MAC address mapping.

```
>st arp

--time: 189.117


  arpRqIn  : 0
  arpRqOut : 15
  arpRspIn : 0
  arpRspOut: 0
  arpRqTossed: 0
  arpRqPased: 0

ARP statistics:

  ARP requests in: 0; out: 15

  ARP replys   in: 0; out: 0

   # MAC Address   i/f pend IP           create_time last_time  IPSTR

   8 000000-000000  1  yes  c900a8c0 110      0       192.168.0.201

Last used arpcache pointer not set
```

18.6.16 802.1x Key Derivation Counters

This command displays counters derived from the operation of the 802.1x key derivation handshake protocol. The number of type of each receved packet is displayed. There are are also counters for Michael and Mic errors.

```
>st sup

--time: 193.115


Wpa State                : WSUP_STATE_WPA_IDLE
802.1x Rcvd              : 0
Msg 1 Rcvd               : 0
Msg 3 Rcvd               : 0
GK Rcvd                  : 0
Msg Timeouts             : 0
Fmt Errs                 : 0
Mich Errs                : 0
Mic Errs                 : 0
```

18.6.17 DHCP Counters

This command provides information about the number of DHCP packets that have been transmitted and received.

```
>st dhcp

--time: 198.119

errors                   : 0
discovers                : 0
offers                   : 0
req                      : 0
acks                     : 0
bpreplys                 : 0
declines                 : 0
release                  : 0
naks                     : 0
renew                    : 0
rebind                   : 0
reject                   : 0
```

18.6.18 "IF" Counters

This command provides information about the number of TCP, UDP, ARP, DHCP, etc.. packets that have been transmitted and received.

```
>st if

--time: 200.856


InOct                    : 0
InUcP                     : 0
InMcP                     : 0
InDis                     : 0
InErr                     : 0
InUnk                     : 0
OutOct                    : 672
OutUcP                    : 0
OutMcP                    : 16
OutDis                    : 16
OutErr                    : 0
```

18.6.19 Deployment Counters

This counters display how many UDP status packets have been sent as well as how many airbeam packets were transmitted/received. The current state of the Airbeam subsystem is also displayed.

```
>st deploy

--time: 205.472


OutUDP                   : 4
InAirb                    : 0
InFTP                     : 300
OutFTP                    : 0
Airbeam state             : 12
```

18.6.20 WTT Counters

This command displays counters associated with the Walkie Talkie subsystem. There are several forms of this command that are selected by the presence or absence of a parameter.

18.6.20.1 Simple form (no parameter)

This form

```
>st wtt

--time: 208.639

pkts tx                   : 0
pkts rx                   : 0
lost 0                    : 0
lost 1                    : 0
lost n                    : 0
```

18.6.20.2 Form 1

```
>st wtt 1

--time: 211.286

pkts tx                   : 0

pkts rx                   : 0
lost 0                    : 0
lost 1                    : 0
lost n                    : 0


TOSS REASONS

No voice packets tossed
```

18.6.20.3 Form 2

```
>st wtt 2

--time: 214.216


pkts tx                   : 0
pkts rx                   : 0
lost 0                    : 0
lost 1                    : 0
lost n                    : 0

wtt gaps


0: 0
1: 0
2: 0
3: 0
4: 0
5: 0
6: 0
7: 0
8: 0
9: 0
```

18.6.20.4 Form 3

```
>st wtt 3

--time: 217.048


pkts tx                   : 0
pkts rx                   : 0
lost 0                    : 0
lost 1                    : 0
lost n                    : 0
```

18.6.21 Unicast Counters

This command displays the counters that record unicast/Private Reply sessions and transmit/received packets. The number of receive packet gaps is also recorded.

```
>st uc

--time: 221.838

uc sessions               : 0
pkt tx                    : 0
pkt rx                    : 0
rx gaps                   : 0
rx timeout                : 0
```

18.6.22 DNS Counters and table

This command displays DNS packet counts, the known DNS servers , and the current name to IP address mappings.

```
>st dns

--time: 225.777


DNS client statistics:


 No DNS servers in list

DNS cache entries:

runtime errors:              : 0
requests sent:               : 0
replys received:             : 0
useable replys:              : 0
dnsc_retry:                  : 0
timeouts:                    : 0
```

18.6.23 Configuration Counters

This command displays information about the state of configuration data base.

```
>st cfg

Flash Statistics\rn

valid sector           : 0
valid items            : 15
invalid items          : 377
valid bytes            : 128
invalid bytes          : 3450
unused bytes           : 4610
```

19 Trace

This command sets the level of trace for the various subsystems on the EWB100 device. There are two forms of the trace state, one for umac tracing and the other for all other subsystem. Changes to trace options take effect immediately.

19.1.1 NonUmac Tracing

For all tracing except umac, the format is:

```
set trace x y
```

Where x is the subsystem name and y is the level of detail. Values for x are:

sys	**vr**	**av**	**led**
ip	**arib**	**bat**	**wsup**
telnet	**wnmp**	**usb**	**diag1**
fsk	**mgt**	**ucos**	**diag2**
scan	**dsp**	**enc**	**hra**
nt	**ui**	**sip**	**cb**
uc	**key**	**flashtest**	
play	**wt**	**kwik**	

Values for y are:

- **none**
- **major**
- **minor**
- **detail**
- **off**

The default for all values is “none”

19.1.2 Umac Tracing

Umac tracing is more complex because there are many tracing entities within the umac subsystem. The general format is:

```
set tra umax xx yy
```

Where xx is the tracing entity and yy is either 0 (no tracing) or 7f (full tracing). There are some other possible values for yy but they are untested.

Values for the tracing entities are listing below. Not all may generate actual tracing output. It is assumed that the user of this command is familiar with the internal structure of the umac software.

all	trap	icache	psmb
ic	chmask	led	psmgt
sm	aes	link	roam
alarm	aloft	llc	sbox
bra	antenna	md5,	sbss
debug	bss	mgt	scan
framework,	cce	mic	sta
global	ccx	mlme	tkip
instance	ckip	mmpdu	toolbox
lmacbuf	compress	mssid	umac
lmacnv	concat	null	wep
lmactmr	db	options	wme
memory	dcf	pdlp	wpa
object	dot11d	phy	vapspoll
pda	dot11h	poll	lpit
pimfor	dpsm	ppe	mtum_cce
pool	fastpath	priv	mtum
rand	frg	privkeys	pattern
shmem	fxs	profiles	tools
table	hibernate	promisc	pdr
timer	hmac	psm	synth

20 MiniKernel Commands

This section describes the minikernel commands. The minikernel cli is entered by entering D (Capital D) immediately after the system boots up. The minikernel will wait a short amount of time (~3-10 seconds, depending on whether the RS232 or USB serial interface is connected) for the D command to be entered before proceeding to start the runtime code.

In addition to the D command, any of the following commands can also be entered. They perform the same function as the “load boot” format listed below but without the need to enter the minikernel CLI

G – start the runtime code

X – upload flash contents using Hyperterm and X-Modem

E- erase the flash that contains all user defined configuration (including user defined defaults)

M – burn the contents of the temporary flash download area into the final flash location. The contents of the temporary flash area contain the final flash area address where it is to be placed.

Once inside the minikernel CLI, there are five available::

- Load boot
- Version

20.1 *Load Boot*

This command controls various aspects of updating flash and controlling the startup of the runtime code. The command sequence is:

```
Load boot xx
```

Where xx can be:

G – start the runtime code

X – upload flash contents using Hyperterm and X-Modem

E- erase the flash that contains all user defined configuration (including user defined defaults)

M – burn the contents the temporary flash download area into the final flash location. The contents of the temporary flash area contains the final flash area address where it is to be placed.

20.2 *Version*

This command displays the version of the minikernel and some hardware information. The command is:

```
version
```

It takes no parameters. The output is:

```
Motorola CA10 Version 1.1.1011 Mini-CA10  Using external mem Apr  8 2010 16:43:42
EngVer = 4 HwVer = 2 nchip = noProt chip = Locosto-Lite
Hawkeye = 0x412b TI_DM = 0x5b66 ES = unknown
```

Appendix A: Status Record Contents

Fields sent out by the EWB100 devices, in order that they appear

Field name	Description	Example value
MAC	Device MAC address	001570DC0EA9
PROFILE	Current user profile	0
BATTERY	Remaining battery	100%
VOL	Volume setting	5.5
PACKAGE	FTP/DA package id	-1601659414
STATUS	Indicates device crashed in past	Not Available (no crash)
VERSION	Runtime software version	Motorola CA10 Version 1.1.1009
HW_MODEL	Device model name	CA1060
HW_VERSION	Device Version	Rev A
SERIAL_NUMBER	Device serial number	9232521100042
MFG_DATE	Device mfg date	08/20/2009
COUNTRY_CODE	Radio operation	WW
PNAM	Profile name	1
WTTTX	Wtt channel	0 (actual channel # -1)
WTTTO	Alternative wtt channel	0 (ditto)
WTTRXMSK	Wtt channels being monitored (bit map)	1
TIME	Time since boot (in us)	3488500
CTIME	Time charging began	3488487
CPLUGIN	Plugged into charger	0x010
BVOLT	Battery voltage	4.050
CCURRENT	Charge current	0
COLOUMBS	Charge amt in coloumbs	0
BTEMP	Battery temp	28
DIETEMP	Die temp	368
CHVOLT	Charge voltage	4.687
USBVOLT	Usb charge volage	0.000
DSPIDLE	DSP idle time	100.00
CLEVEL	Charge level	100%
OCV	unknown	4.045
RADIOON	Radio is on	0
AUDIOON	Audio is on	0
CCYCLES	# of battery charge cycles	30.2
APMAC	Current AP mac adr	01570dee71
CHAN	Current AP channel	11
SIGNAL	RSSI	40
RTC	Current time	141073
AID	AID number	4
CAPINFO	AP capabilities	0
ASSOCTIME	Time of last	11733

	association (ms)	
ASSOCTED	Associated	1
TRAPSI	Internal counter	5
ASSOC	Associations	2
REASSOC	Reassociations	0
LASTAP	Previous AP mac address	000000
LSTARTS	Locally started wtt sessions	0
RSTART	Remotely started wtt sessions	0
RDECLN	Remote wtt sessions declined	0
RI	Remote wtt sessions ignored	0
LTO	Listen timeout value	0
RXTXCNT	Wtt rx/tx transitions	0
RXTXWN	Wtt sessions “won”	0
RXTXLST	Wtt sessions “lost”	0
PKTTX	Wtt tx packets	0
PKTRX	Wtt rx packets	0
LOST0	Wtt rx zero packet gaps	0
LOST1	Wtt rx 1 packet gaps	0
LOSTN	Wtt rx > 1 packet gaps	0
UROAMS	802.11 roams	1
USCANS	Active scans	0
UASSOC	Association	1
DEAUT	Deauthentication pkts rcvd	0
MISBEA	Missed beacons	12015
DSSlpBlkng	Low power Development counter	0x0
DSAttempt	ditto	0x2640
DSReady	ditto	0x89
DSSlept	ditto	0x89
DSGsmAwake	ditto	0x15
DSKeyAwake	ditto	0x0
DSUmacActive	ditto	0x74
DSDspIdleMode	ditto	0x0

Appendix B: Utility Program

This section describes how to use a number of “utility programs” that are useful with the EWB100

B.1 clientencrypttool

There is a Windows program for calculating encrypted values for WPA and CLI configuration values. This program can be found in the EWB100 section of http://support.symbol.com. The program runs from a Windows Command Line. The syntax is:

cliencryptool xxx

where xx is the value to be encrypted.

The program will calculate the encrypted form and print it out as a series of hex digits, separated by a colon. The string, minus the colons, should be entered into the CLI. Note that the encrypted form will be much longer than the input string and will always contain only hex digits and be a multiple of 32 characters.

Also note if that if the following special characters are used in a password:

& | ( )

Then they must be escaped with the special character ^. The reason for this is that the Windows Command Line interpreter uses these four characters for its own purposes and so will not pass them to the program. By escaping them, they will be passed to cliencryptool.

Appendix C: Utilizing the USB Interface with Win/XP

This appendix describes how to install the EWB100 USB Drivers and how to use Hyperterm to communicate with the EWB100 CLI when using Windows XP, 32 bit edition. The USB drivers can be found in the EWB100 section of http://support.symbol.com. Vista/Windows 7 setup is described in a later section.

The EWB100 uses a special physical connector that plugs onto the side of the device. The device interface is also used by the headset connector. Only the USB connector or headset connector may be used at any given time. The USB connector has a mini-USB interface that can be used with a compatible USB cable. There is one such connector included within each EWB100 charging station.

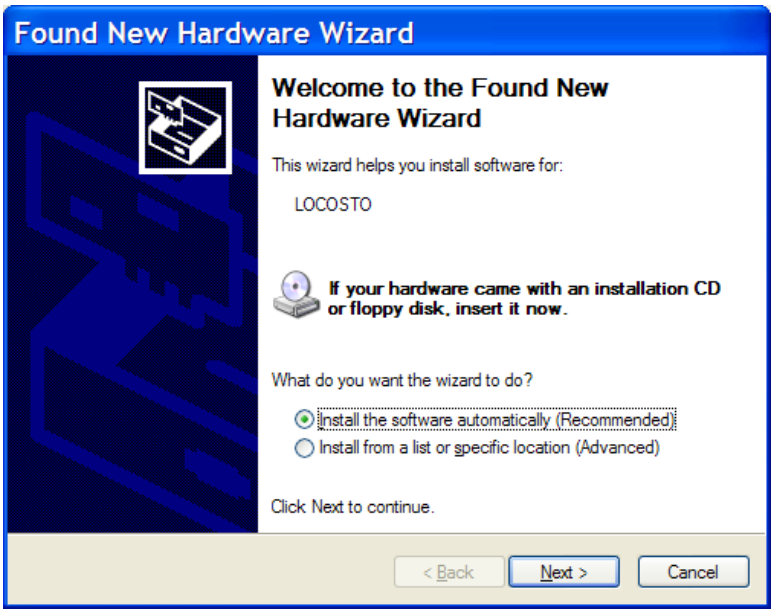
There are three USB endpoints contained in the EWB100. The first is embedded in the bootup firmware in the CPU chipset used on the EWB 100. This endpoint has the label “LOCOSTO” and should be ignored. The second endpoint is part of the “Mini-Kernel” and can generally be ignored unless the user is trying to access the Min-Kernel. The third endpoint is the EWB100 runtime code. Both the Mini-Kernel and Runtime endpoints are called “CA10” and use the same Windows drivers.

The “CA10” is an older model number for the device now known as the EWB100 and is embedded in the EWB100 USB software.

Which endpoint is used depends on the state of the EWB100 device. When the device is powered on, the LOCOSTO endpoint is available for only a few seconds. Normally users will not see this endpoint. The exception to this is when a powered down device is plugged into the USB interface on a PC. If this is done it will cause the device to power on and then the LOCOSTO endpoint will be available for a short time.

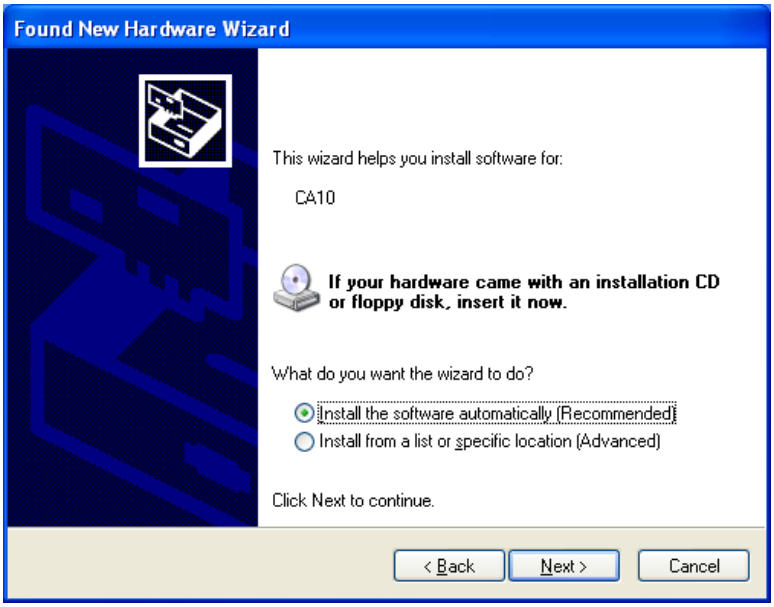
If a powered down device is plugged into a PC, one will see the following window being displayed. The user should just cancel it and close the window.

IMPORTANT: It is likely that the user will need to repeat the driver installation process for each physical USB interface on the PC that the EWB100 is plugged into. Normally it will need to be done only once per interface but if a USB hub is inserted between the EWB100 and the PC, it may need to be repeated.

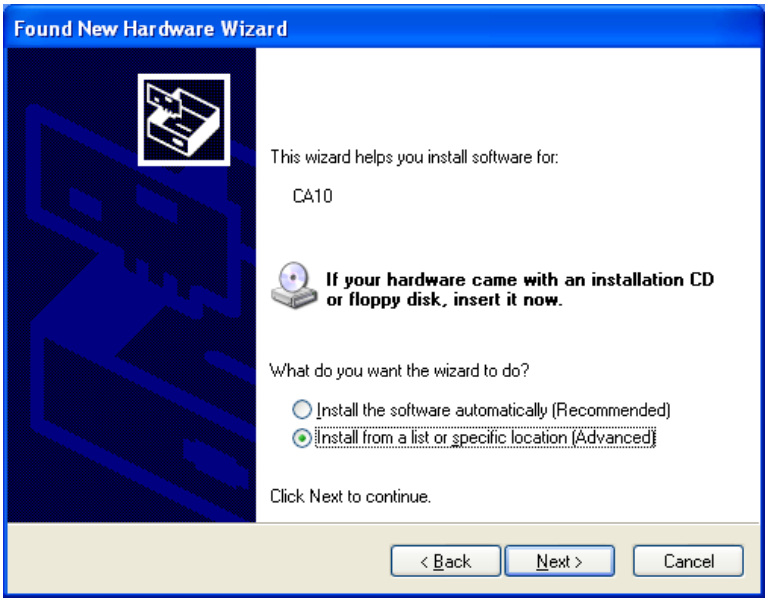


The following window sequence will appear after the above window is closed. It will also appear if a running EWB100 is plugged into a PC for the first time.

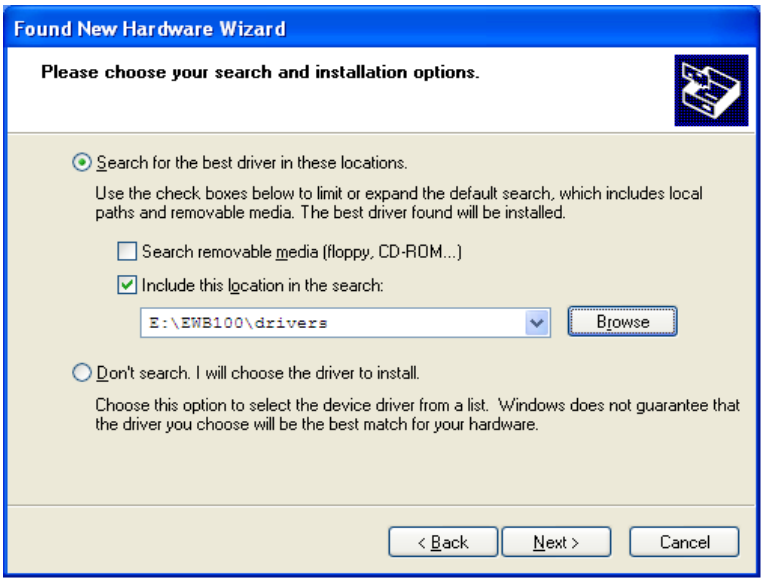
Select Next



Select “Install from a specific location” and select next



Then select "Include location in the search"

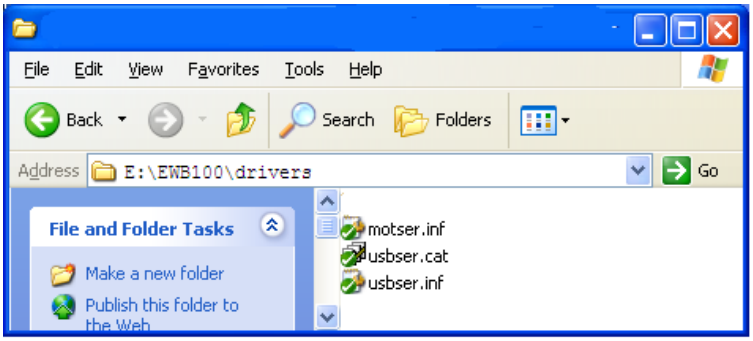


This assumes the drivers are located in an external drive/directory named:

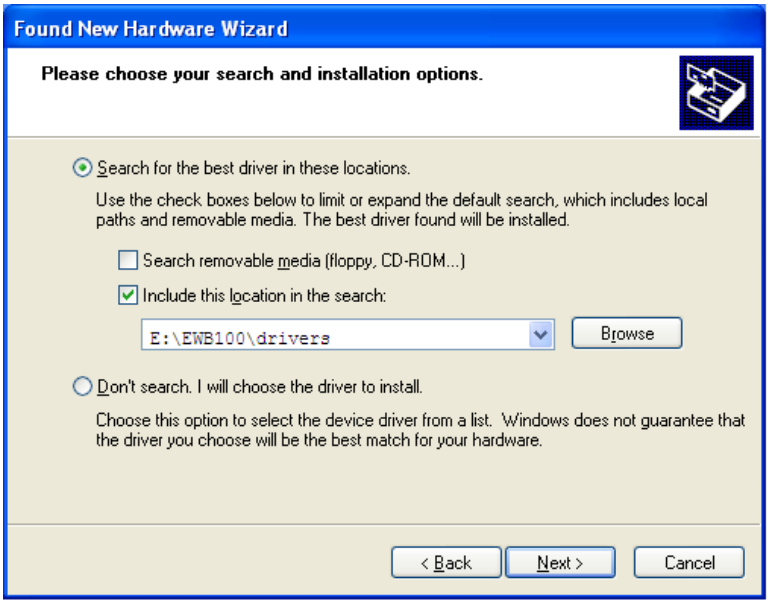
E:\EWB100\drivers

The contents of this directory are:

.

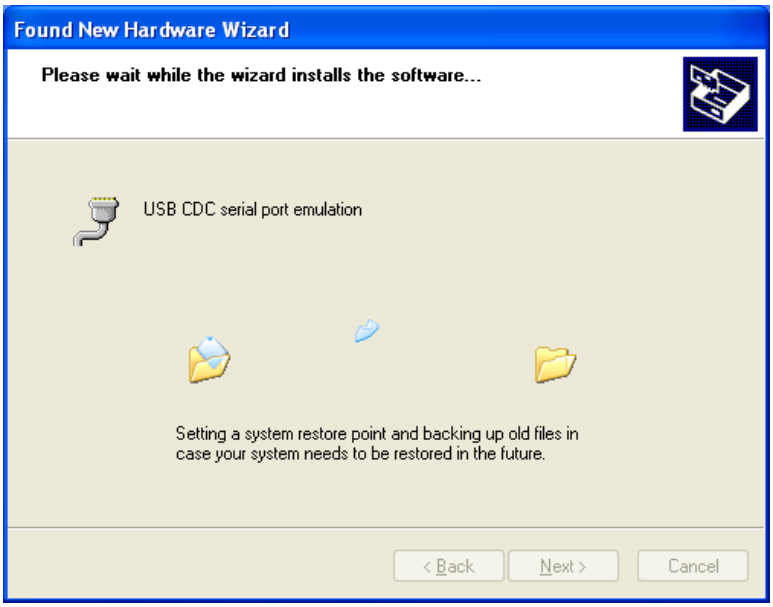


Enter this directory in the search box as shown below and select Next.

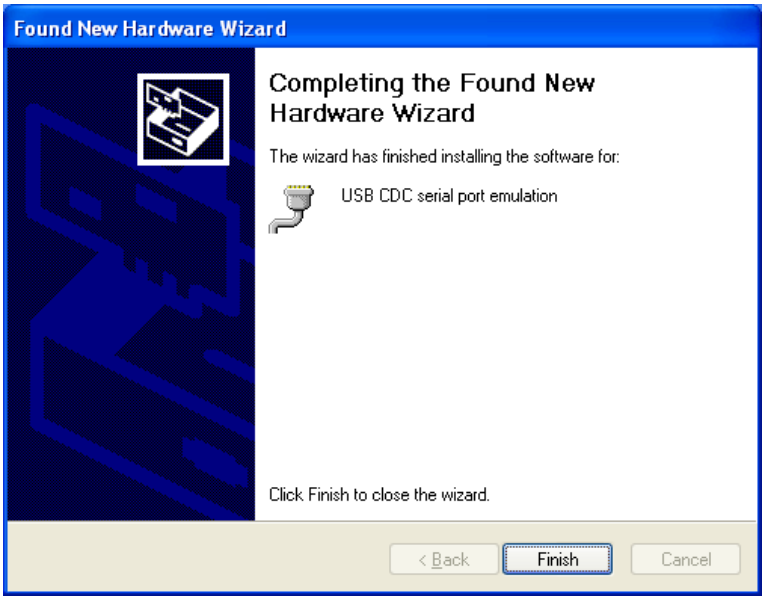


Wait while the driver installs

Wait



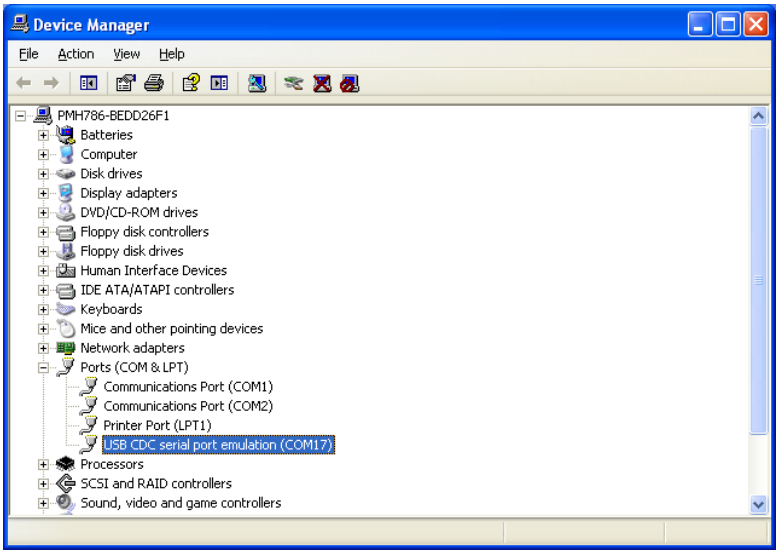
The following indicates success



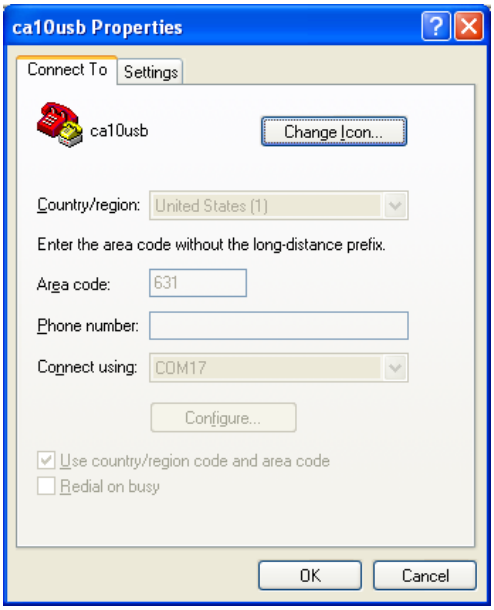
At this point the driver is installed and we can startup Hyperterm.

The USB connection will appear as a serial port to Hyperterm. The exact port number will generally depend on which physical USB port on the PC the EWB100 is plugged into as well as the presence of other USB peripherals. One can determine the assignment by looking at the “Device Manager” window

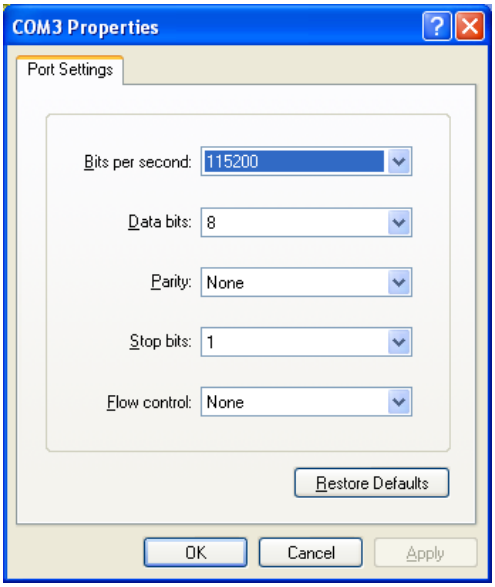
available from the “Control Panel” as seen below:



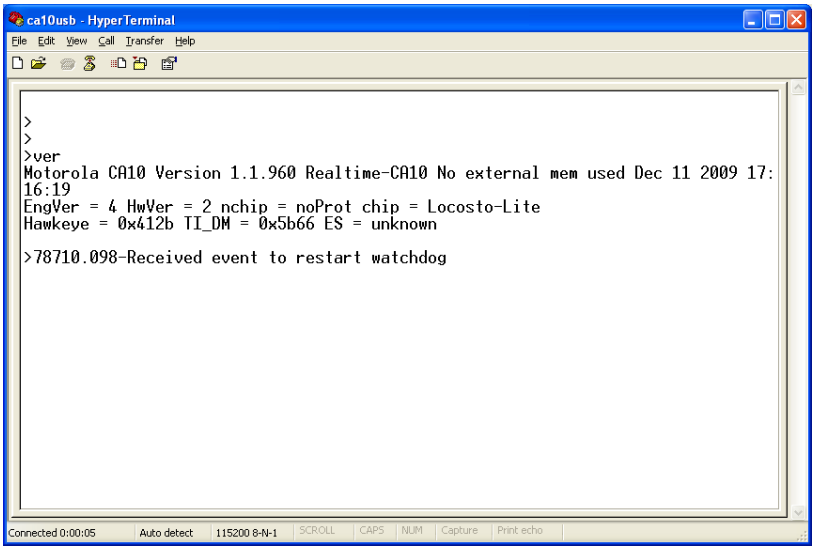
Alternately one can wait and just open the “properties” menu item on Hyperterm and see the list of possible ports. In any case one must pick the desired port to connect with via this window.



One must also configure the Port settings. The settings should be set as seen below:



At this point “Connect” to the port and you should see approximately the following. Alternatively one may just see the CLI prompt “>”. How much one sees depends upon where the EWB100 is in its bootup sequence when the USB is attached to Hyperterm.



Appendix D: Utilizing the USB Interface with Vista and Win/7

This section will describe how to use the USB interface with Windows Vista and Windows 7 (64 bit versions). This process uses the sames USB drivers as used with Windows XP. We assume they are on the external drive/directory: E:\EWB100\drivers. The USB drivers can be found in the EWB100 section of http://support.symbol.com.

Prior to proceeding users should first read Appendix C on utilizing the USB interface with Windows XP in order to understand how the various USB endpoints on the EWB100 operate. The EWB100 endpoints operate identically with all versions of windows.

It should be noted that Hyperterm is no longer included with either Windows Vista or Windows 7. One may purchase a copy from:

http://www.hilgraeve.com/

Alternatively there are a number of free terminal emulators available from:

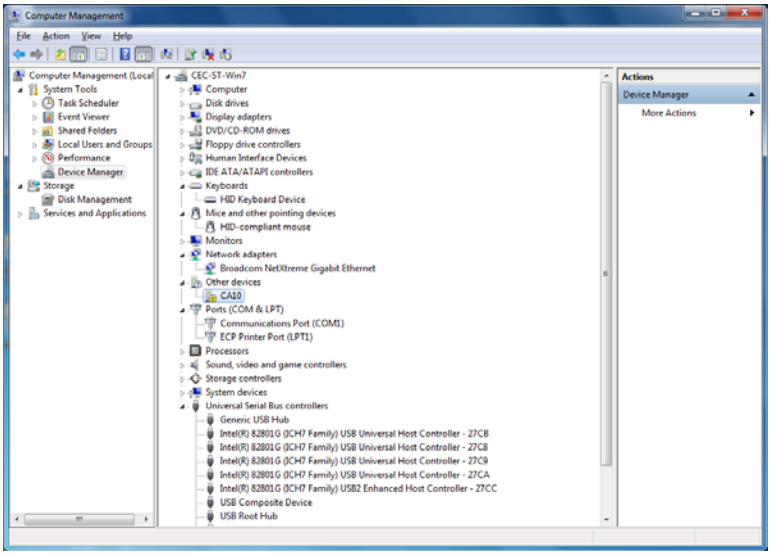
http://www.recursion.jp/prose/hypoterm/

http://www.putty.org/

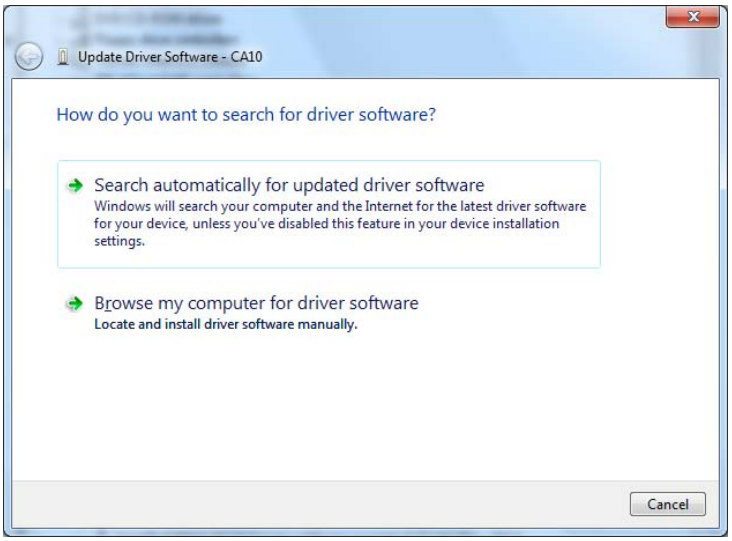
To install the drivers on Windows Vista/Win7, follow this procedure:

Plug a running EWB100 into the Vista/Win7 computer.

If the “Add New Hardware” window does not appear, Right click on My Computer->Manage. The following window will appear:

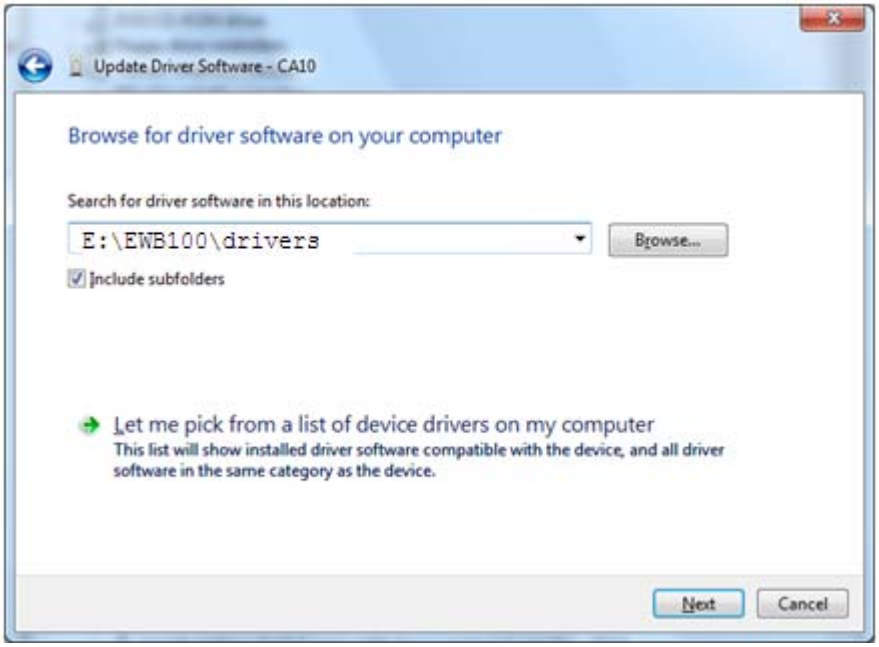


Right click on “CA10” and select “Update Drivers”. The following window will appear:

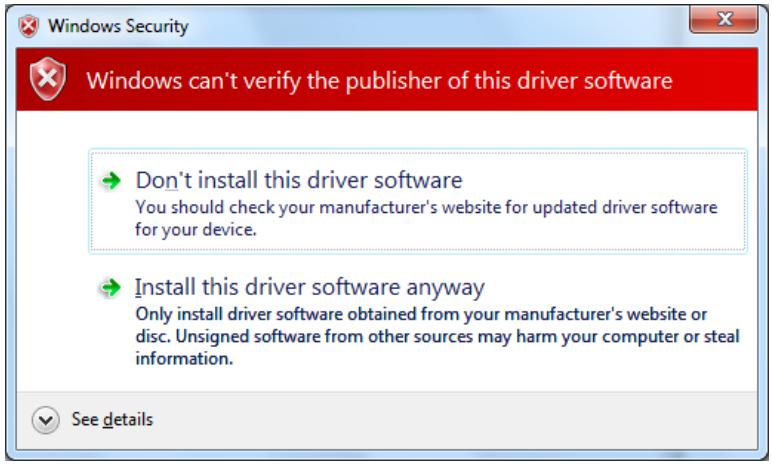


Select the 2nd option – “Browse my computer for driver software”

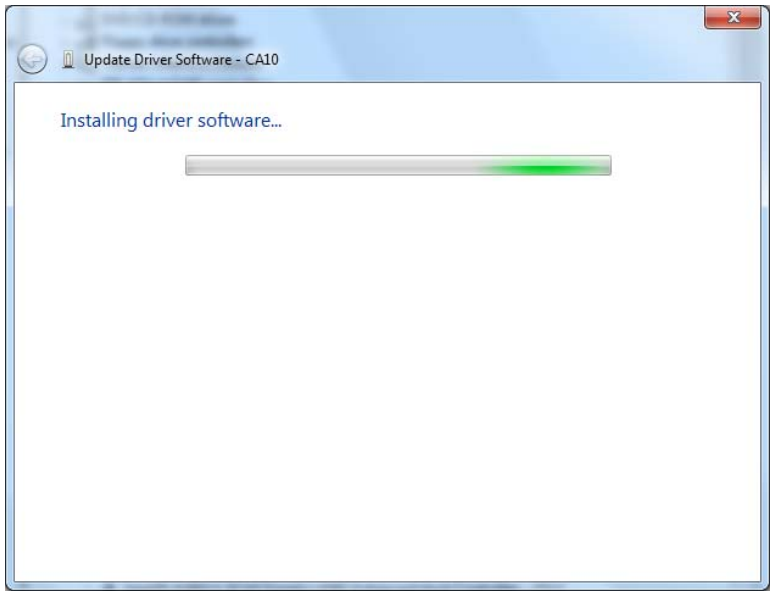
Enter the location where you have copied the files for the USB CA10 drivers.



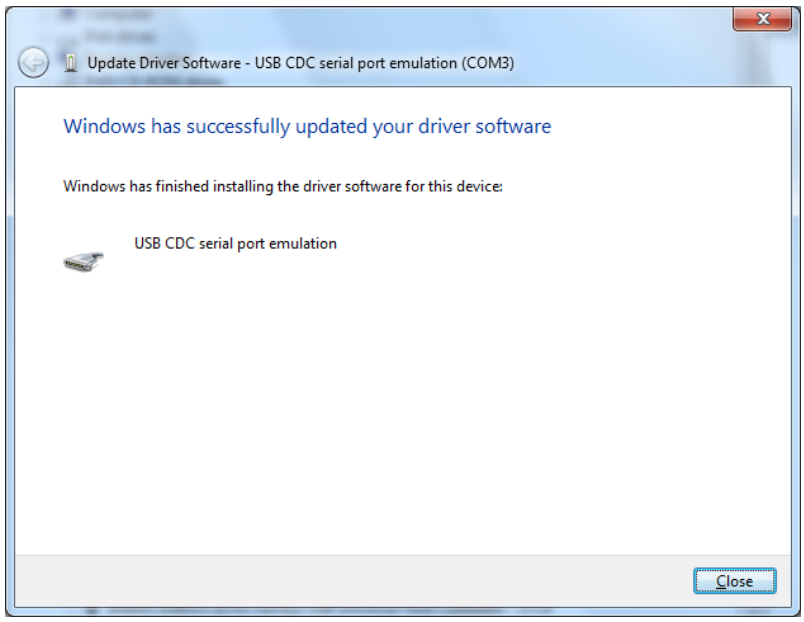
Point to the expanded folder with the CA10 drivers (E:\EWB100\drivers) and select it. Ignore the unsigned driver warning.



Win7 will now install the software:



And the completed pop-up:



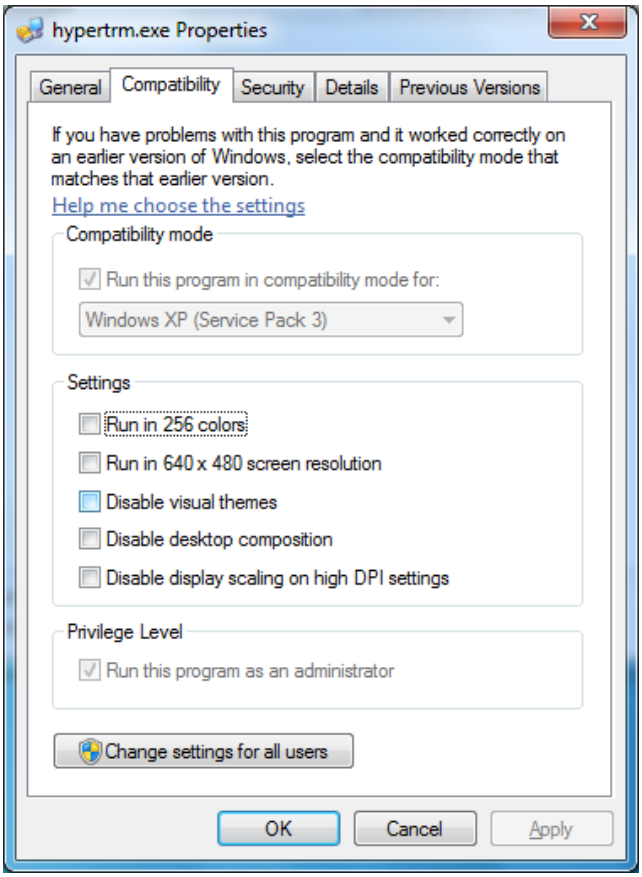
At this point the drivers will have been installed. Remove the EWB100 and reinsert it. The drivers should recognize the device now.

As noted above, inserting the device into a different physical USB port on the PC may require reinstalling the drivers.

The terminal emulator program, such as hyperterm, can be configured to use the newly installed drivers. If hyperterm is used, the configuration procedure described in Appendix C may be used. If another emulator is used, the configuration process may differ. In any case, always select the following parameters:

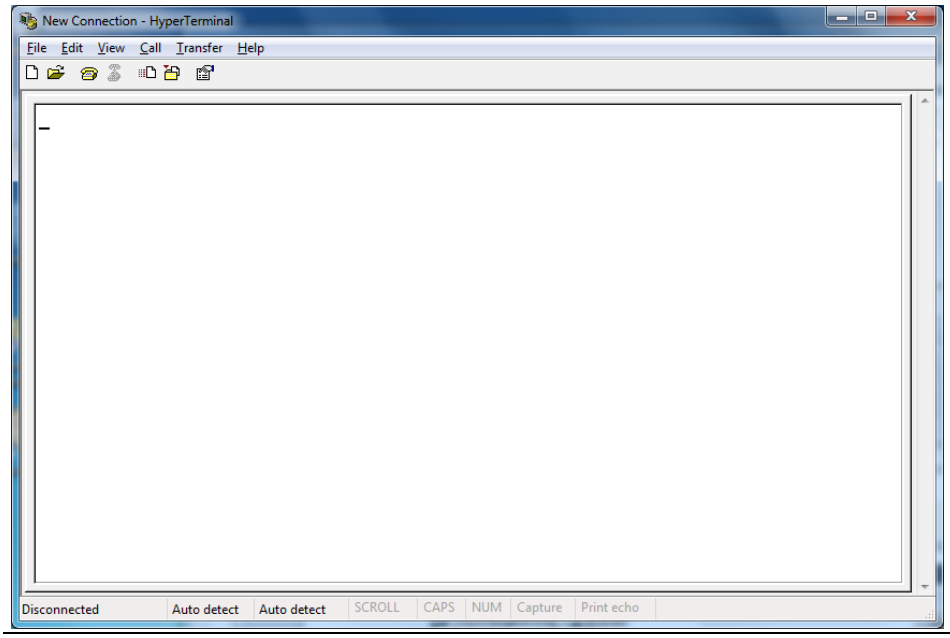
- 115K baud
- No parity
- 8 bit data, 1 stop bit

When using some terminal emulator programs, such as Hyperterm ported from Windows XP, it is necessary to run the program in WinXP-SP3 compatibility mode and as administrator (Win7 pro 64bit). If this is not done, the CA10 will be continuously reset by Windows. This process is shown below for Hyperterm. In order to set the permissions properly, right click on the executable file and select Properties. This will open the following window:



A final note in operation with Win7: After installing the drivers, but before plugging in device to use with hyperterm, have hyperterm running and ready to open the device as you plug in the EWB100/CA10. A pre-saved profile can be created to facilitate this:

Use file->open to select your previously saved connection.



While the device is connected in this fashion with hyperterm executing in admin mode, you should not experience any resets due to Win7 of the device.

Appendix E: Default Dictionary for Audio Prompts

silence

zero

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

30

40

50

60

70

80

90

100

hundred

thousand
million
billion
trillion
unknown
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
small
plus

add
minus
subtract
times
asterisk
star
slash
divide
pound
percent
percentage
underscore
at
colon
question_mark
period
comma
not
semicolon
ampersand
less
greater
equals
than
to
exclamation_point
battery
signal
good
fair
poor
roam
retries

beacons
missed
excellent
priority
ip
mask
address
gateway
default
dot
dhcp
static
silence1
mac_address
associated
ess
bss
ap
security
open
wep
wpa
key
index
volume
scanner
symbologies
packets
bytes
cfg_error
and
or
enable

disable
sku
dollar
dollars
cent
cents
yes
no
today
yesterday
tomorrow
next
week
month
year
this
last
ready
notready
please
repeat
understood
okay
fail
special
inventory
number
outofstock
instock
onorder
failed
quantity
on

hand
delivery
discount
return
store
other
enter
item
off
profile
monday
tuesday
wednesday
thursday
friday
saturday
sunday
server
system
AM
BOTTOM
DIRECTORY 0
DIRECTORY 1
EIGHTIETH
EIGHTEENTH
ELENTH
PM
PROFILE0
PROFILE1
directory

urgent
previous

only
high
low
front
back
dozen
aisle
endcap
head
room
side
bunch
group
pounds
strength
lower
of
hundredth
first
second
third
fourth
fifth
sixth
seventh
eighth
ninth
tenth
channel
frequency
version
hex
volt

volts
udp
twentieth
twelfth
thirteenth
thirtieth
tcp
temperature
sixteenth
sixtieth
seventeenth
seventieth
shelf
shelves
port
coulombs
degree
degrees
celsius
batteries
ounce
thousandth
nineteenth
ninetieth
fourteenth
fifteenth
critical
usb
charging
try_to_associate
registers
sales
managers

floor
expert
novice
wpa2
aes
tkip
calling
everyone
duplicate
manager
asstmgr
pharmacy
auto
operator
asstmgr1
asstmgr2
again
are
available
call
do
found
from
register
registered
say
successfully
want
you
incoming
actions
active
begin

bob
break
button
clean
click
complete
completed
create
created
department
device
double
end
forward
front
george
idle
item
items
list
login
logout
mark
mary
nancy
pause
reassign
release
reply
resume
select
side
single

talk
todo
tom
user
users
waiting
task
tasks
main
menu
accepted
started
using_channel
100pc
75pc
50pc
25pc
nosignal
useralert1
useralertn
wttalert1
wttalertn
checkin
runtime
started
multiple
valid
inprogress
tts
condition
staging